

# 基于同构多核处理器的 H.264 多粒度并行编码器

于俊清 李 江 魏海涛

(华中科技大学计算机科学与技术学院 武汉 430074)

**摘 要** H.264 码率低和视频质量高的优越性能以增加编码计算的复杂度为代价,如何开发适用于多核处理器平台的并行编码算法是提高其编码速度的重要研究内容,对于满足高清视频实时传输和大规模共享具有十分重要的意义.利用 H.264 开源编码器项目 X264,在片级和数据级并行编码算法的基础上,通过分析图像帧之间的参考关系,提出并实现了 B 帧个数可变的帧级并行算法;根据宏块之间的参考关系,设计了一种类似流水线的宏块级并行方法;基于 Intel 同构多核平台,提出融合帧级、片级、宏块级和数据级 4 种不同粒度的并行编码方案,开发了 H.264 多粒度并行编码器.实验结果表明,在码率增加不大的情况下,H.264 多粒度并行编码器可以很好地提升编码加速比,视频编码质量符合高质量的要求.

**关键词** 多核处理器;多线程;H.264 编码器;多粒度并行

中图法分类号 TP391 DOI号: 10.3724/SP.J.1016.2009.01100

## Multi-Grain Parallel H.264 Encoder for Homogeneous Multi-Core Architectures

YU Jun-Qing LI Jiang WEI Hai-Tao

(College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074)

**Abstract** As a new generation video coding standards, H.264 requires more intensive computation than before to achieve high coding performance. With the continuous development of multi-core processors, the development of the multi-grain parallel H.264 encoder has great significance to meet the requirements of video real-time transmission and large-scale sharing. Based on the open-source H.264 encoder project X264, an adaptive B-frame frame parallel algorithm is proposed through analysis of the dependency among the referent frames. According to the reference of related macroblocks, a macroblock-level parallel approach is designed in assembly-line fashion. A multi-grain parallel H.264 encoder is implemented on the homogeneous multi-core platform. This encoder efficiently combines four parallel grains, including frame, slice, macroblock and data level. Experimental results demonstrate that the encoding speedup is improved to a large extent, without obviously increasing bitrates. The high-quality video is kept in the encoding process.

**Keywords** multi-core processor; multi-thread; H.264 encoder; multi-grain parallel

## 1 引 言

由于网络的迅猛发展,人们对视频数据的共享

和应用越来越广泛,如网络电视、视频会议等需求的不断涌现,对视频编解码速度和播放质量都提出了很高的要求.如今的视频编码标准已经向着低码率高质量的方向发展,H.264 视频编码标准是这种发

展趋势的典型代表. H.264 能以较低的数据速率传送基于 IP 的视频流,在视频质量、压缩效率和数据包恢复等方面,都超越了现有的 MPEG-2、MPEG-4 和 H.26x 视频编码标准,更适合窄带传输. 在相同的重建图像质量下,H.264 比 H.263 节约 50%左右的码率<sup>[1-2]</sup>. 然而,H.264 获得的优越性能是以增加计算复杂度为代价的,标准中加入了很多新的编码特性,这些编码特性在相同图像质量情况下大幅压缩码率的同时,编解码复杂度也随之大大增加,从而造成编解码速度很慢. 据估计,编码计算复杂度大约相当于 H.263 的 3 倍,解码计算复杂度大约相当于 H.263 的 2 倍. 目前,支撑并行化的软硬件环境有了很大的发展,其中同构多核处理器的迅速发展为并行程序提供了很好的平台<sup>[3]</sup>. 因此,开发多核环境下的 H.264 多粒度并行编码器是提高其编码速度的重要研究内容,对于满足高清视频实时传输和大规模共享具有十分重要的意义.

本文根据以往研究中存在的问题,基于开源项目 H.264 编码器 X264,在其原有片级和数据级并行编码实现的基础上,提出并实现了一种 B 帧个数可变的帧级并行算法和类似流水线的宏块级并行算法,通过有效融合帧级、片级、数据级和宏块级 4 种不同粒度并行方法,运用 POSIX 多线程库,在同构多核平台下成功开发了 H.264 多粒度并行编码器.

本文第 2 节将对 H.264 并行编码的现有研究成果进行分析和总结;第 3 节在分析 H.264 层次结构的基础上讨论其可并行粒度,重点阐述帧级、片级、宏块级和数据级并行解决方案并给出相应的实现方法;第 4 节通过实验测试多粒度编码器的性能;第 5 节总结全文工作并对未来工作进行展望.

2 相关工作

目前已经有一些研究机构在从事 H.264 并行编解码的研究工作. Intel 研究中心提出了 Intel 超线程体系结构下的 H.264 并行编码模型<sup>[4-6]</sup>. 对于帧级并行,他们对共用参考帧的可编码帧分片之后实施并行,从其实验结果可以看出,编码加速比有显著的提高,但这种帧级并行方法局限于 B 帧数目固定的情况,而且是通过片级并行来间接实现,编码器中多种并行粒度的融合程度还不高. Intel 中国研究中心和浙江大学的研究人员分别运用 SIMD 指令对 H.264 离散余弦变换<sup>[7]</sup>和解码<sup>[8]</sup>进行了并行化研究,主要是运用了单指令多数据(Single Instruction

Multiple Date, SIMD)的向量运算特性,对 H.264 计算量比较集中的变换和量化部分进行改写. 加利福尼亚大学的研究人员提出了一种宏块级并行的算法. 它利用重建宏块和当前正编码宏块对多个宏块进行并行处理,并利用官方 JM 编码器进行了一些简单的实验测试<sup>[9]</sup>. 中国台湾的研究人员讨论了 H.264 宏块编码中运动估计和运动补偿方法中的并行<sup>[10-13]</sup>. 但是这些并行更适合片上系统(System On a Chip, SOC)的并行,对于通用计算机来说,由于频繁的传递数据会造成很大的负载.

通过对现有研究成果的分析可以看出,现有 H.264 并行编码研究中,片级和基于多媒体扩展指令集的数据级并行研究成果最多,实现方法比较成熟;研究者也提出了一些帧级并行编码思路 and 实现方法,但研究尚不够充分;对于宏块级并行算法,宏块级并行尚没有十分成熟的实现方法. 除了 Intel 中国研究中心的研究员提出了帧级并行和片级并行融合之外,绝大部分研究都是针对 H.264 编码单一粒度的并行. 针对以上研究中存在的问题,本文将设计和实现一种灵活性高的 B 帧个数可变的帧级并行算法和类似流水线的宏块级并行算法,并与片级和数据级并行编码算法进行融合,实现 H.264 的多粒度并行编码器,实现在保证编码视频质量的基础上,达到尽可能大的编码性能.

3 H.264 多粒度并行编码方法

3.1 并行粒度的划分

H.264 编码层次结构按照从大到小可以分为图像组、帧、片组、片、宏块和子块. 如图 1 所示,这些结构之间是一种层层包含的关系,根据每个层次不同的特点,可以设计不同的方法来提高编码效率. 下面分别介绍各个层次的特点,并对每个层次中存在的并行性进行分析.

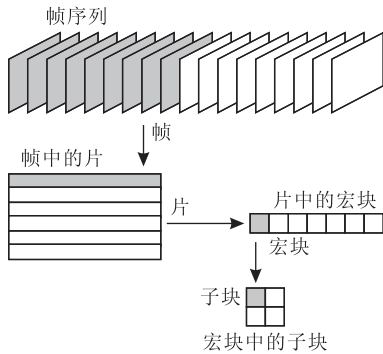


图 1 H.264 层次结构示意图

### 3.1.1 图像组

在 H.264 中,帧图像分为 I、P、B 3 种,一般排成 IBBPBBPBBPBBPBBP... 样式,这种连续的图像组合即为图像组(Group Of Picture,GOP),是 H.264 视频存取的基本单位.

由于 GOP 是 H.264 最基本的存储单位,GOP 之间不存在相互关系,符合并行化的一个条件,但由于 GOP 数据量太大,不能作为处理器基本处理单位进行整体处理,因此不能在这个层次实施并行化.

### 3.1.2 帧结构

H.264 视频将图像(即帧)分为 I、P、B 3 种,I 帧是帧内编码图,P 是预测图,B 是双向预测图.简单地讲,I 帧是一个完整的图像,而 P 帧和 B 帧记录的是相对于 I 帧的变化,没有 I 帧、P 帧和 B 帧就无法解码.虽然当前编码帧依赖于前面的已编码帧,但通过分析帧间的依赖关系可以找到共用参考帧的当前编码帧,这就决定了可以在帧级实现并行编码.

### 3.1.3 片组和片结构

一个帧图像可编码成一个或很多个片,每片包含整数个宏块,即每片至少一个宏块,最多时每片包含整个图像的宏块.设片的目的是为了限制误码的扩散和传输,应使编码片相互间保持独立.某片的预测不能以其他片中的宏块为参考,这样某一片中的预测误差才不会传播到其他片中去.

H.264 编码中片的特点是帧中的片之间互不依赖,也就是说,如果一幅帧图像被划分成  $N$  个片,这  $N$  个片之间彼此不存在相互依赖关系,这便为实现同一帧中片的并行奠定了基础.

### 3.1.4 宏块结构

一帧编码图像通常划分成若干个宏块,每个宏块由一个  $16 \times 16$  的亮度像素块和附加的一个  $8 \times 8$  Cb 和一个  $8 \times 8$  Cr 彩色像素块组成.每个图像中,若干宏块被排列成片的形式.

宏块作为 H.264 编码最小的基本单元,其预测模式选择需要参考左、上、右上的相邻宏块,在所需参考宏块均已编码的情况下,多个宏块就可以实施并行预测模式分析和编码,其具体的并行处理流程类似流水线操作.

### 3.1.5 子块结构

在 H.264 的运动预测中,一个宏块可以被分为不同的子块,形成 7 种不同模式的块尺寸,即分成  $16 \times 16$ 、 $16 \times 8$ 、 $8 \times 16$  或  $8 \times 8$  亮度像素块;如果选了  $8 \times 8$  的子块,则可再分割成各种子块,其尺寸为  $8 \times 8$ 、 $8 \times 4$ 、 $4 \times 8$  或  $4 \times 4$  亮度像素块.由于 H.264

编码的最小基本单元是宏块,因此子块一级的编码包含在宏块的内部,因此无法实施并行操作.

通过以上分析可以看出,要想实现 H.264 多粒度并行编码器,在现有层次结构中,按照并行粒度从大到小可以分为帧级、片级和宏块级.现有多核处理器通常都有多媒体扩展指令集,利用 SIMD 指令集对向量操作同样可以实施数据运算的并行,能够获得可观的加速比.因此,H.264 多粒度并行编码包括帧级、片级、宏块级和数据级 4 种不同的粒度.

## 3.2 帧级并行

在 H.264 编码中,I 帧是内部编码帧,不需要参考其它帧,P 帧需要前向的 I 帧作为参考,B 是双向预测帧,需要前向和后向的 I 或者 P 帧作为其参考帧.由于帧与帧之间的参考关系比较复杂,彼此之间互相关联,对帧编码的简单并行是行不通的.因此,寻找共用参考帧的可编码帧成为实现帧级并行的关键.

假设编码序列中设置的 B 帧个数为 2,其具体视频编码序列为 IBBPBBPBBP...,依照 I、P 和 B 帧之间的参考关系,可以把连续的视频序列按照 BBP 的样式分割成一个个单元序列.经过分析可以看出一个 BBP 单元序列中的两个 B 帧由于共用前后的两个 P 帧作为参考帧,可以实施并行.同时,这个 BBP 单元序列中的 P 帧又作为下一个 BBP 单元序列中 P 帧的参考,因此前一单元序列中的两个 B 帧加上下一个单元序列中的 P 帧就可以实施三帧同时并行.以上是 B 帧设置参数为 2 时帧级并行的基本思路,可以得出原本执行一帧的时间现在可以用来执行三帧,理论加速比基本可以达到 3,等于 B 帧设置参数加 1.

本文实现的帧级并行可以随着 B 帧设置参数动态变化,具体的并行策略如图 2 所示.当设置编码序列中 B 帧个数可变时,帧级并行的线程数取决于 B 帧的个数,B 帧数越大,并行加速比越高,在处理器足够的情况下,理论上获得的最大加速比等于 B 帧设置参数加 1.由于在并行过程中,创建编码线程



图 2 帧级并行策略示意图

需要分配内存,线程之间的数据传输也会消耗资源,实际加速比会小于理论加速比,再加上处理器个数资源的限制,会在某个 B 帧的个数上获得一个峰值加速比。

3.3 片级并行

H.264 编码的片与片之间不存在直接的数据依赖,可以通过对帧进行分片来实现多片同时编码,从而实现片级并行。片级并行的实现通过把帧图像数据单元平均划分成多个数据块,然后每个数据块同时创建线程进行并行编码,编码过程中会加入片头信息,编码完成后再依次把各片数据组合成一帧图像数据就完成了。具体的片级并行编码流程如图 3 所示。

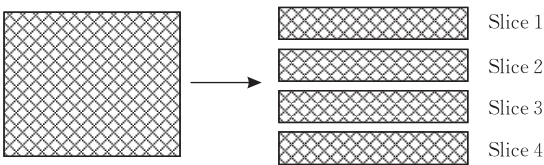


图 3 片级并行策略示意图

片级并行程度取决于帧的分片数,而分片数可以人为设定,分片数越大,获得的并行加速比越大。但是由于片是人为把图像帧划分成的数据块,而划分之后片与片之间相关性将不复存在,因此,原本存在于帧内部的数据相关性,可以很好地用于宏块预

测并可以提高压缩比,现在就无法利用,造成单位时间内压缩视频的规模(即码率)增大。码率是 H.264 编码性能的一个重要参数,因此,不能以无限制地提高分片数目来获得加速比,需要综合考虑分片数目对码率的影响。

3.4 宏块级并行

片由多个宏块组成,宏块编码需要左、上和右上的宏块进行参考,串行编码程序中片内的宏块按照行顺序依次进行编码,当前宏块进行编码的时候其参考宏块已经编码完成了。宏块级并行就是要找到参考宏块已经编码完成的可编码宏块。

宏块级并行主要针对片内有多行宏块的情况,首先创建两个线程分别对第 1 行宏块和第 2 行宏块同时进行编码。由于宏块编码过程中要参考左、上和右上的宏块,因此,第 2 行宏块的编码肯定要比第 1 行宏块编码慢,大概相差两个宏块编码的时间。当第 1 行宏块编码结束之后,第 2 行的宏块也编码到了最后几个宏块,这样第 3 行的宏块能以第 2 行的宏块为参考,可以直接进行编码。因此,当第 1 行宏块编码完成之后,宏块编码线程就可以直接跳转到第 3 行进行编码;同理,当第 2 行编码完成之后,宏块编码线程 2 就直接跳转第 4 行进行宏块编码,依次类推。具体的宏块级并行策略如图 4 所示。

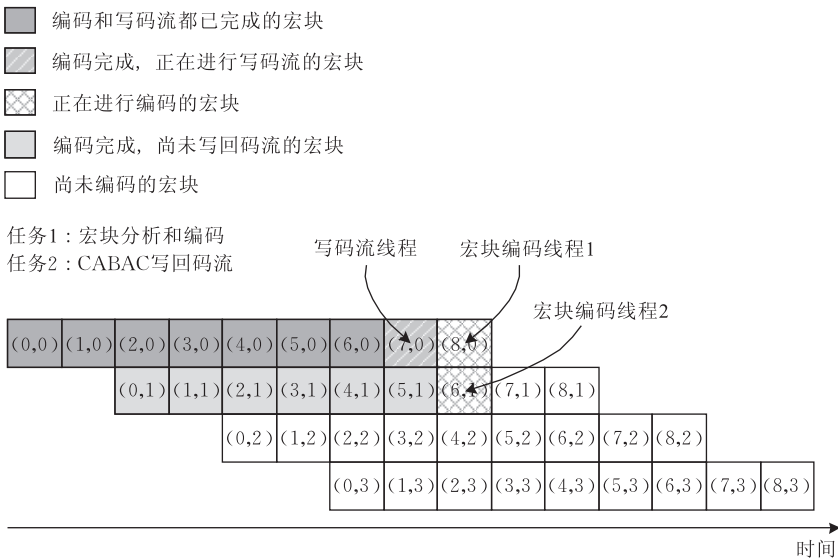


图 4 宏块级并行策略示意图

3.5 数据级并行

同构多核处理器内嵌了多媒体指令集,这些指令集通过对数据实行向量化运算从而达到并行处理的效果,多媒体指令集包括 MMX/SSE/SSE2,它们对大规模的密集计算有明显的加速作用。SIMD 向

量运算模式如图 5 所示。

在 X264 代码中,主要运用多媒体指令集对如下几个计算模块进行改写。具体包括离散余弦变换、运动补偿、去方块滤波、1/4 或 1/8 像素精度的运动矢量计算以及量化。由此获得的数据级并行加速比



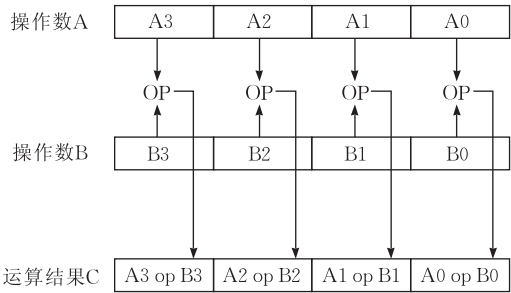


图 5 SIMD 向量运算模式示意图

也是很可观的,数据级并行使用的是处理器内嵌的多媒体指令集,不占用多处理器的资源,是一种非常有效的提高编码速度的并行策略,同时可以把节省下来的处理器资源用于大粒度的编码并行化。

3.6 多粒度并行编码实现

完整的 H.264 多粒度并行编码实现流程如图 6 所示. 基于 X264 开源编码器,通过创建 3 个数据队列,分别为 I/P 编码帧队列、B 编码帧队列和 NAL 输出缓冲队列,并建立分级线程实现。

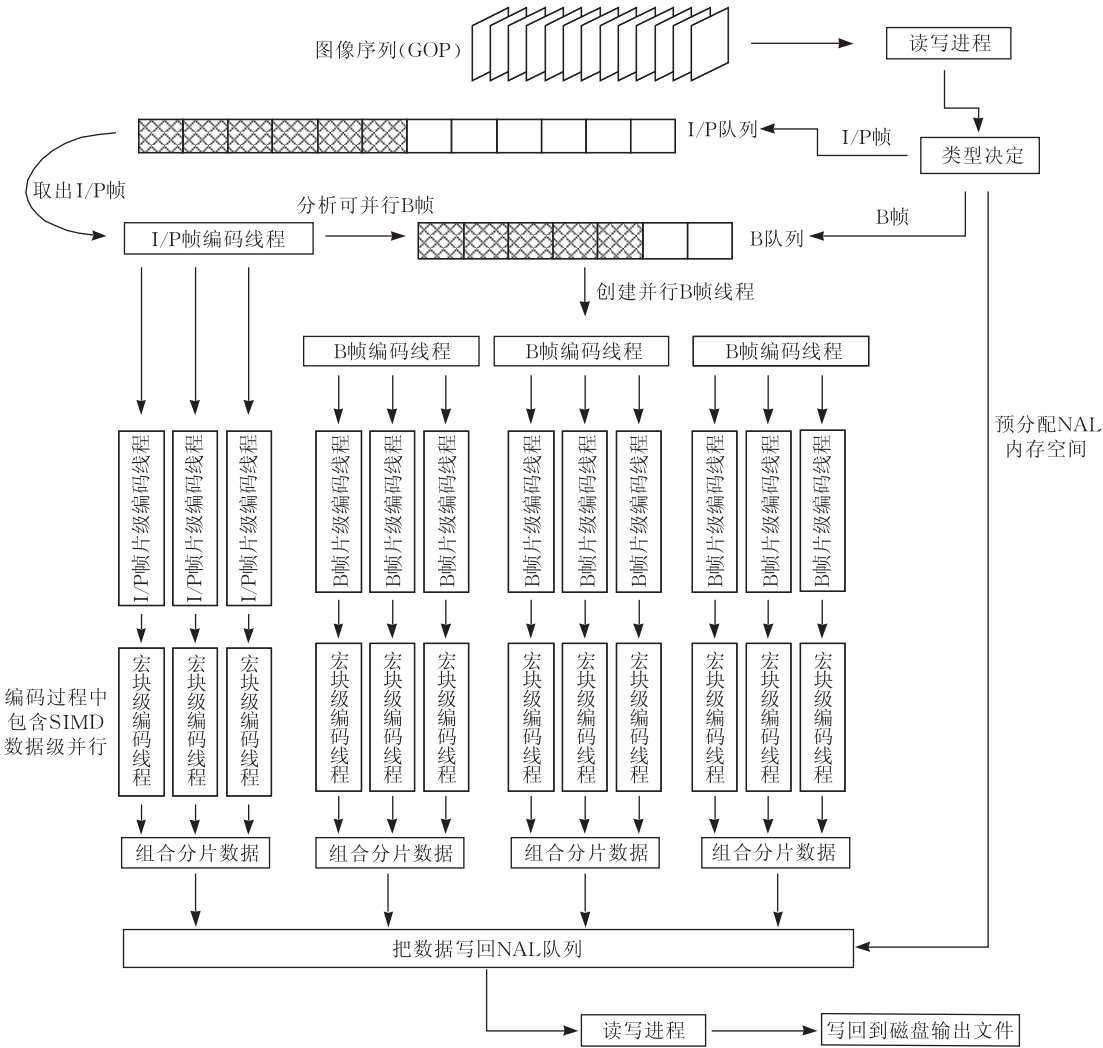


图 6 H.264 多粒度并行编码实现示意图

H.264 编码过程分成 3 个部分:原始视频输入、编码和编码后数据输出. 其中涉及的线程有主线程、读写线程、I/P 帧编码线程、B 帧编码线程、片级编码线程和宏块级编码线程. 这些线程创建的先后关系及主要功能为:主线程创建读写线程和 I/P 帧编码线程;读写线程负责从缓冲队列中读入图像序列、决定帧类型并插入到编码帧队列中,同时检查 NAL 队列把完整数据写回到磁盘空间;I/P 帧编码线程

进行帧级并行分析,创建 B 帧编码线程;I/P 帧编码线程和 B 帧编码线程分别根据分片数创建片级编码线程;在各片内部,分别创建两个宏块级编码线程和一个码率写回线程对宏块进行类似流水线的并行编码. 下面详细阐述多粒度并行编码实现的算法流程。

(1) 原始视频序列被读写线程顺序读入到内存中并决定帧类型. 如果是 B 帧就放入 B 帧编码队

列,否则放入 I/P 帧编码队列中,同时要在 NAL 输出队列中为编码后的码流输出预留一个数据单元,在该数据单元中写入图像帧的帧号.

(2) 主线程在程序初始化时创建 I/P 帧编码线程,该线程通过轮询的方式不断检查 I/P 帧编码队列中是否有可编码的 I 或 P 帧. 如果存在,则再查找 B 帧编码队列,把所有共用参考帧的可并行 B 帧全部取出来,为所有取出的 B 帧创建 B 帧编码线程与取出的 I 或 P 帧一起实行帧级并行编码.

(3) 在帧级并行线程创建之后,如果处理器核还有剩余,根据分片数的大小,把每一个并行编码帧再次进行分片,为每一个片创建片级编码线程以实现片级并行.

(4) 在每个片内部,创建两个宏块级编码线程和一个宏块码率写回线程,编码线程 1 和线程 2 分别对片内奇数行和偶数行的宏块顺次进行编码,由于宏块之间的参考关系,两个线程相互交替刚好达到流水线并行的效果. 两个编码线程把编完的码流依次写入到各自的缓冲队列中,宏块码流写回线程分别读两个缓冲队列把宏块码流按顺序进行组合.

(5) 在取宏块级编码过程中,宏块编码内部的离散余弦变换和量化等都可以运用 SIMD 多媒体扩展指令集实施数据级并行.

(6) 片级编码结束之后,通过把各分片数据加入片头信息组合成帧码流,然后加入帧头信息,根据 NAL 队列预留数据单元中写入的帧号,由读写线程把完整帧码流写回到 NAL 队列中与该帧帧号相同的数据单元中去.

从上述多粒度并行编码算法流程可以看出,帧级、片级、宏块级和数据级 4 种粒度的并行可以很好地融合在一起. 这些并行粒度之间有一个大致的顺序,一般先有帧级并行再有片级并行,然后是宏块级并行,在宏块编码内部还包括数据级并行. 在具体的编码执行过程中,也可以根据计算环境关闭其中的某几个级别的并行粒度,例如可以关闭片级和数据级并行只使用帧级和宏块级并行.

下面通过伪代码详细描述整个编码流程中的两个重要线程:读写线程和 I/P 帧编码线程.

读写线程代码:

```
while (there is frame to encode)
{
    if (there is free entry in image buffer)
    {
```

```
        read a frame to the buffer;
        decide the type of current frame;
        if (the type is I or P)
            enter I/P queue;
        else
            enter B queue;
        allocate a node in the NAL queue for current frame;
    }
    else if (there is bitstream in the NAL queue)
        write the bitstream to output file;
    else
        wait;
}
```

I/P 帧编码线程代码:

```
while (1)
{
    if (there is frame in the I/P queue)
    {
        fetch a frame from I/P queue;
        analyze the B frames in B queue;
        create B thread for B frame which can be encoded in parallel;
        encode current frame;
    }
    else if (all frames are encoded)
        exit;
    else
        wait;
}
```

## 4 实验结果与分析

### 4.1 实验平台和参数设置

我们采用的同构多核实验平台为 Intel Xeon 2×4 核处理器,单核主频 2.0GHz,8.0GB 内存. 实验主要测试多粒度融合、B 帧个数自适应之后的加速比和码率的变化情况以及多粒度并行融合对不同分辨率和不同格式视频的加速比变化情况. 为了便于在相同测试条件下进行比较,主要测试参数设置如下:30 帧一个图像序列、多参考帧、可以平均 B 帧参考块的运动矢量、允许 B 帧加权预测、关闭自适应 B 帧判定.

### 4.2 多粒度并行编码实验结果分析

H.264 多粒度并行编码共包括 4 个粒度的并行,本节首先对帧级、片级和数据级并行加速比与传统单一级并行性能进行分析,然后加入宏块级并行,分析多粒度并行编码的加速比和码率变化.

4.2.1 帧级、片级和数据级并行与传统单级并行性能的分析比较

图 7 给出了帧级和片级两种粒度并行编码的融合与传统单一的帧级、片级并行加速比的对比情况. 在这里, 设定 B 帧数为典型值 2. 从图 7 中可以看出随着分片数的增加, 片级并行加速比会逐渐增大, 范围大概在 1~4 之间变化, 并在分片数为 6 或 7 时达到峰值, 这主要是由处理器核个数的限制造成的. 从图中可以看出, 帧级并行加速比不随分片数的变化而变化, 因为帧级并行只取决于帧之间的参考关系. 由于 B 帧参数设置为 2, 其理论加速比为 3, 除去创建线程、数据传输的资源消耗, 实际的加速比约为 2. 当帧级和片级融合之后, 与文献[4-6]相比, 加速比有显著提升, 范围从 2~5 变化, 同样在分片数为 6 或 7 的时候达到加速比峰值. 由帧级和片级的并行方案可以得出, 帧级并行只需要与 B 帧设置参数相同的处理器核的个数就能获得较好的加速比, 而片级并行需要处理器核的个数较多的时候才能获得较大的加速比.

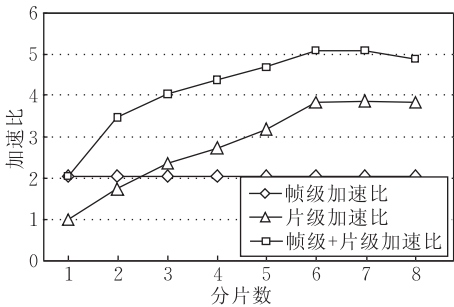


图 7 帧级和片级并行融合与传统单一的帧级、片级并行加速比对比示意图

图 8 是在图 7 的基础上加上数据级并行, 实现了 3 个粒度的融合, 曲线变化趋势与图 7 大致相同, 但是整体加速比大幅提高, 3 种粒度融合之后加速比变化范围从 20~35. 与文献[4-6]单一地采用固定帧级和片级并行编码相比, 加速比提高了 5~7

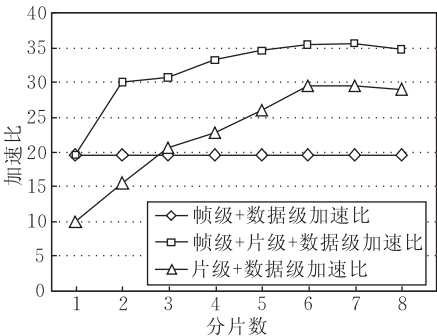


图 8 帧级、片级和数据级并行融合加速比示意图

倍. 由此可见, 数据级并行对于提高 H.264 编码速度具有非常重要的意义. 由于数据级并行只需要处理器支持 SIMD 多媒体扩展指令集, 而不需要占用多个处理器核资源, 因此是一种高效的并行编码方法.

4.2.2 宏块级并行加速比和编码码率分析

图 9 给出了 B 帧个数可变时多粒度并行编码时码率的变化情况. 由于分片是把帧图像划分成多个数据块, 隔断了片与片之间宏块的相关性, 使得宏块之间的参考只能局限于片内部, 因此分片数越大, 码率越大, 由此造成了片级并行编码码率随分片数线性增大. 帧级并行的码率随着 B 帧数的增加先降低而后增加, 在 B 帧个数为 2~4 时取得较小值, 这是由于 B 帧数较小时, 双向预测的相关性大, 压缩率较大, 随着 B 帧数增加双向预测的相关性变小, 从而造成了码率的增加. 而宏块级和数据级并行并不影响相关性, 因此对编码码率都没有影响. 可以看出, 随着分片数的不断增大, 多粒度并行码率基本上呈现线性增大的趋势. 当 B 帧个数为 2~4 的时候, 整体码率相对于其它 B 帧个数比较低.

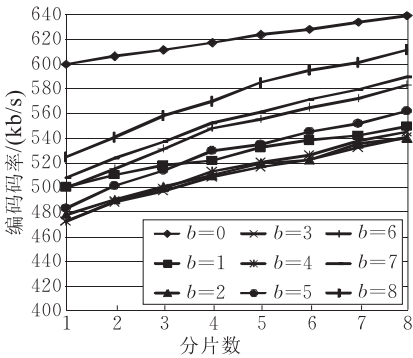


图 9 B 帧个数可变时多粒度并行码率变化

图 10 和图 11 给出的是 B 帧个数可变时帧级、片级并行和帧级、片级及宏块级并行加速比的对比状况. 从图中可以看出, 随着 B 帧个数的不断增大, 并行编码加速比都呈上升趋势. 结合图 9 的码率变化趋势图可以看出, B 帧数设置为 2~4 的时候编码码率较低, B 帧数设置大于 3 的时候加速比在一个较高的水平, 由此可以得出存在一个 B 帧个数最优值使得编码速度较高和编码码率较低. 从实验结果可以大致看出这个点出现在 B 帧设置个数为 4 的时候.

图 10 和图 11 的对比可以看出, 帧级片级融合时峰值加速比出现在分片数为 6 的地方, 而帧级、片级和宏块级并行融合时峰值加速比出现在分片数为 3 的地方, 二者峰值加速比大致相当. 加入了宏块级

并行之后,峰值加速比出现在分片数较小的地方,根据图 9 码率的变化趋势可以得出,加入了宏块级并行在大幅提升编码速度的同时,编码码率相对较低,这是宏块级并行的优势所在.

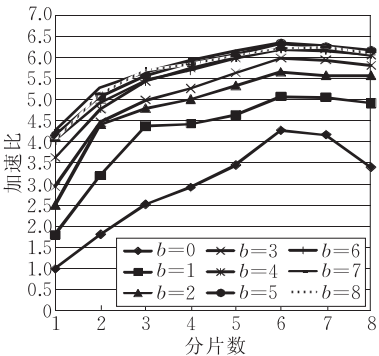


图 10 B 帧个数可变时帧级片级融合加速比

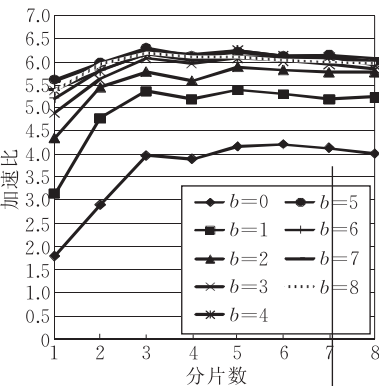


图 11 B 帧个数可变时帧级片级宏块级融合加速比

4.2.3 与现有其它方法的性能比较

如表 1 所示,我们从加速比和码率比两个方面将本文的多粒度并行方法与现有其它方法进行性能比较.文献[4-6]在 8 个逻辑 CPU 上(4CPU 的 Xeon 超线程处理器),即对于 B 帧为 2 的视频采用片级并行取得 4.31x~4.69x 加速比,当分片数为 8 时,该方法引起了 1.08 倍的码率增加.在 4.1 节介绍的测试平台中,本文对其方法进行了测试,获得 5.21x~5.56x 加速比.文献[4,8]和文献[9]分别采用了数据级和宏块级的并行取得了 2x~4x 和 3.08x 的加速比,同时保持了码率不变.

通过对多粒度并行融合,我们的多粒度 H.264 并行编码获得 33.57x~34.36x 的加速比,比上述方法有较大的性能提高.同时与文献[4-6]相比,通过宏块级并行部分替代了片级并行,减小了分片数,从而在取得较高加速比的同时,只带来了 1.04 倍的码率增加,与文献[4-6]相比,保持了较低的码率.

表 1 本文方法与其它方法的性能比较

并行策略	加速比	码率比
B 帧固定片级并行 <sup>[4-6]</sup>	5.21x~5.56x	1.08x
单一数据级并行 <sup>[4,8]</sup>	2x~4x	1x
单一宏块级并行 <sup>[9]</sup>	3.08x	1x
本文的多粒度并行	33.57x~34.36x	1.04x

4.3 不同视频格式下实验结果分析

视频格式按分辨率从大到小排序为 CIF(352×288)>SIF(352×240)>QCIF(176×144).如图 12 所示,可以看出分辨率越大,融合之后获得的编码加速比越大.这是由于分辨率越大,一帧图像中需要的计算时间也就越多,相对于帧级线程创建和内存分配的消耗,并行编码中计算所占的权重比较大,通信开销所占的权重相对较小,也就会获得更大的加速比.QCIF 分辨率大概是 CIF 的四分之一,其加速比相对于 CIF 也小很多.因此,可以得出结论,不同视频格式下多粒度并行编码融合之后加速比都有不同程度的提升,多粒度并行对于分辨率大的视频编码提升的效果更为明显.

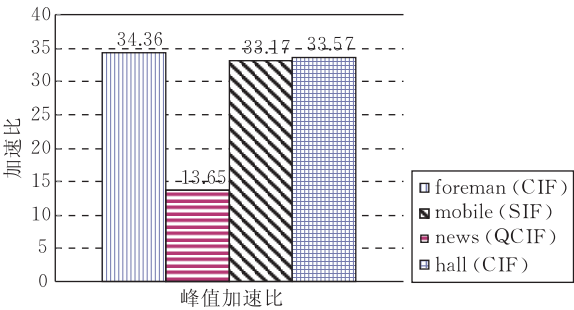


图 12 不同视频格式下多粒度并行编码加速比对比

5 结 论

H.264 并行编码的粒度主要包含帧级、片级、宏块级和数据级.单一粒度的并行化可以增加加速比,但是往往有加速比不够高或者码率增大的弊端.本文实现的 H.264 多粒度并行编码器,对帧级、片级、宏块级和数据级并行进行了融合,通过减少分片数和降低码率在保持视频质量(PSNR)不变的同时<sup>[6]</sup>,极大地提升了并行编码的加速比,获得了优越的编码性能.因此,基于同构多核处理器实现的 H.264 多粒度并行编码器是一种综合性能优异的并行编码器.对于多粒度并行来说,在处理器核有限的情况下,如何在各个粒度上更精确地分配数目不同的核资源是优化编码性能的主要因素之一,未来工作将围绕这一点进行研究.



## 参 考 文 献

- [1] Wiegand T, Sullivan G J, Bjontegaard G et al. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 2003, 13(7): 560-576
- [2] JVT. Draft text of final draft international standard for advanced video coding. Geneva: ISO/IEC, Draft: ISO/IEC 14496-10, 2002
- [3] Parkhurst Jeff, Darringer John, Grundmann Bill. From single core to multi-core: Preparing for a new exponential//*Proceedings of the IEEE International Conference on Computer-Aided Design*. New York, 2006: 67-72
- [4] Ge S, Tian X, Chen Y-K. Efficient multithreading implementation of H.264 encoder on Intel hyper-threading architectures//*Proceedings of the IEEE Pacific-Rim Conference on Multimedia*. Singapore, 2003: 469-473
- [5] Tian X, Chen Y-K, Girkar M et al. Exploring the use of hyper-threading technology for multimedia applications with Intel OpenMP compiler//*Proceedings of the Intel Parallel & Distributed Processing Symposium*. Nice, 2003: 36-43
- [6] Chen Y-K, Tian X, Ge S et al. Towards efficient multi-level threading of H.264 encoder on Intel hyper-threading architectures//*Proceedings of the 18th Intel Parallel and Distributed Processing Symposium*. Santa Fe, 2004: 63-72
- [7] Ye Jian-Hong, Liu Ji-Lin. Fast parallel implementation of H.264/AVC transform exploiting SIMD instructions//*Proceedings of the International Symposium on Intelligent Signal*

*Processing and Communication Systems*. Xiamen, China, 2007: 870-873

- [8] Zhou X, Li E Q, Chen Y-K. Implementation of H.264 decoder on general-purpose processors with media instructions//*Proceedings of the SPIE Conference on Image and Video Communications and Processing*. California, 2003. Bellingham: SPIE-IS&T, 2003: 224-235
- [9] Zhao Zhuo, Liang Ping. A highly efficient parallel algorithm for H.264 video encoder//*Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. Toulouse, 2006: 489-492
- [10] Fluegel S, Klusmann H, Pirsch P et al. A highly parallel sub-pel accurate motion estimator for H.264//*Proceedings of the IEEE 8th Workshop on Multimedia Signal Processing*. Victoria, BC, 2006: 387-390
- [11] Lin Chichun, Lin Yukun, Chang Tiansheuan. A parallel multi-resolution motion estimation algorithm and architecture for HDTV sized H.264 video coding//*Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. Hawaii, 2007: 385-388
- [12] Cho C-Y, Huang S-Y, Wang J-S. An embedded merging scheme for H.264, AVC motion estimation//*Proceedings of the IEEE International Conference on Image Processing*, Barcelona, Spain, 2003: 909-912
- [13] Lee Chuanyiu, Lin Yucheng, Wu Chiling et al. Multi-pass and frame parallel algorithms of motion estimation in H.264 AVC for generic GPU//*Proceedings of the IEEE International Conference on Multimedia and Expo*. Beijing, 2007: 1603-1606



**YU Jun-Qing**, born in 1975, Ph.D., associate professor. His research interests include digital media processing and retrieval, multi-core programming environment.

**LI Jiang**, born in 1984, M. S.. His research interest is digital video processing.

**WEI Hai-Tao**, born in 1982, Ph. D. candidate. His research interest is multi-core programming environment.

## Background

As a new video coding standard, H.264 developed by Joint Video Team (JVT) is now being used widely due to its high-quality video content and low bitrates. A number of new features have been introduced in H.264 to attain such coding performance, such as multiple reference frames, intra-prediction, quarter-pixel motion estimation (ME) with variable block sizes, CABAC, in-loop deblocking filter and more which make encoding process more complex and require more

computation power than previous coding standards. Therefore, we need some hardware and software mechanisms to accelerate the speed of encoder for real-time application.

Multi-core processor architecture is becoming the mainstream solution for next generation microprocessor chips, in which multi-grained parallel computational resources are provided. Multi-core hardware can be used to execute multiple thread contents in parallel. In order to improve H.264 enco-