多媒体基准测试程序中的流并行性分析

周丽萍[1] 安 虹[1,2] 徐 光[1] 王耀彬[1]
(中国科学技术大学计算机科学与技术系 合肥 230026)[1]
(中国科学院计算机系统结构重点实验室 北京 100080)[2]

摘 要 在分析多媒体基准测试程序Mediabench特征的基础上,以Imagine流处理器为例讨论了流体系结构对多媒体应用所提供的软硬件支持,并且利用流编程模型对多媒体应用中存在的流并行性进行了详细的剖析,最后通过对3个典型的多媒体应用进行流并行程序设计,在Imagine的时钟精确模拟器Isim上测试得到了多媒体应用在流体系结构上可以获得的加速性能。

关键词 多媒体基准测试程序,流体系结构,流并行性
中图法分类号 TP338 **文献标识码** A

Stream Parallelism Analysis for Multimedia Benchmark

ZHOU Li-ping[1] AN Hong[1,2] XU Guang[1] WANG Yao-bin[1]
(Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China)[1]
(Key Laboratory of Computer System and Architecture, Chinese Academy of Sciences, Beijing 100080, China)[2]

Abstract We analyzed the characteristics of multimedia benchmark Mediabench and discussed the stream architecture how to provide hardware and software support for multimedia applications taking Imagine stream processor as example. Then we made use of stream programming model to analyze the stream parallelism existed in multimedia applications and select three representative applications for programming stream applications. Through making experiment on Isim simulator, we got the speedup performance of multimedia application working on Imagine processor.

Keywords Multimedia benchmark, Stream architecture, Stream parallelism

1 引言

多媒体应用在日常生活中的地位日益重要,涉及范围也日益广泛。如何能够快速有效地执行媒体处理程序成为学术界和工业界关注的焦点。目前支持多媒体处理的手段包括3种:(1)媒体加速部件,即某些专用芯片;(2)通用处理器上的多媒体扩展指令集;(3)可编程媒体处理器。专用芯片虽然处理性能很高,却只针对某些特定的应用;多媒体扩展指令由于受到通用处理器性能的限制,也只能处理有限的、计算量较小的应用;可编程媒体处理器结合了二者的优点,既可达到很高的处理性能,又有可编程以及可适应不同媒体处理要求的特点。流体系结构是一种新兴的可编程媒体处理器体系结构,它结合了向量和超长指令字体系结构的特点,能有效地加速媒体应用的执行,其中有许多关键技术问题仍未解决。

本文以典型的具有流体系结构特征的Imagine流处理器[1]为例,使用gprof工具和多媒体基准测试程序Mediabench[2]针对多媒体应用进行了详细的剖析,讨论了流体系结构对多媒体应用的软硬件支持,并且利用流编程模型对多媒体应用中的流并行性进行了深入的挖掘,最终通过实验的方法取得了典型多媒体应用在流体系结构上的并行性能。

2 流体系结构对多媒体应用的支持

2.1 多媒体基准测试程序Mediabench的典型特征

Mediabench是第一个集中了多媒体和通信系统所有应用程序的基准簇,包括从图像处理、通信和DSP领域中精选出来的19个应用程序,抓住了多媒体和通信应用程序的基本特征。图1给出了Mediabench包含的测试程序。

从图1中可以看到,Mediabench覆盖了多媒体应用的各个方面。测试程序的计算量主要集中在多媒体运算,包括视频、声音、图像、文字等的编码解码、分析处理、显示转换等。Fritts小组[3]曾经从多个方面对Mediabench进行了深入的分析,得到了媒体处理程序对体系结构提出的要求。

领域	测试程序	主要数据类型	简要说明
Video	MPEG2	byte4	视频压缩程序,主要算法为DCT,IDCT,Rle,huffman编码等
Audio	ADPCM	int	自适应差分脉冲压缩算法

到稿日期:2008-06-30 本文受国家自然科学基金重点项目(60633040),国家973计划项目(2005CB321601),国家863重大项目(2006AA01A102-5-2),教育部-英特尔信息技术专项科研基金项目(MOE-INTEL-08-07)资助。
周丽萍(1982-),女,硕士生,主要研究方向为并行编程模型,E-mail:fancyzlp@mail.ustc.edu.cn;安 虹(1963-),女,博士,副教授,主要研究方向为并行计算机体系结构、微处理器芯片体系结构;徐 光(1982-),男,博士生,主要研究方向为流处理器体系结构;王耀彬(1982-),男,博士生,主要研究方向为并行编程模型。

Graphics	Mesa	float	OpenGL 的一个 3-D 图像库的克隆
Image	JPEG	byte4	标准表态图像压缩算法
	EPIC	float	基于一种直角的塔式小波快速分解算法以及 run-lenght/Huffman 结合的信息编码
	Ghostscript	int	对附注的解释程序，单一应用 gs 针对文件 I/O
Speech	GSM	float	利用 RPE-LTP 混合编码将 160 个 13-bit 的样本桢压缩成 260-bit
	G. 721	short	使用 ADPCM 算法的语音压缩协议
	Rasta	float	支持 PLP，RASTA 和 Jah-RASTA 3 种技术的语音识别程序
security	PGP	float	使用消息摘要来形成签名，RSA 算法
	Pewit	float	公开密钥加密和鉴定程序

图 1 Mediabench 的组成

1)浮点操作相对较少，算术运算操作的频率远大于其他操作，访存操作比较频繁，最终对各功能单元的需求比例为(ALU，mem，branch，shift，FP，mult)⇒(4，2，1，1，1，1)。

2)基本块的大小为 8，并且静态分支预测和动态分支预测的预测效果相差无几，即多媒体应用中控制流的走向是极有规律的，可静态预测。

3)输入数据集较大，工作集较小。实验数据显示 32kB 的数据 cache 可以捕获 76.1%的空间局部性，8kB 的指令 cache 可以捕获 86.8%的空间局部性。

4)程序结构化，形成了一种流数据的处理方式，即处理器持续地加载一个小的数据集，在其上执行所有必要的操作，然后将结果输出，而最初所使用的那些数据将不再会使用到。

显然，媒体处理的特征有别于一般的应用程序，必须有针对性地提供软硬件支持，才能有效地利用媒体处理的特征，使其高效地执行。

2.2 流体系结构对多媒体应用提供的软硬件支持

本文以具有典型流体系结构特征的 Imagine 流处理器为代表，讨论流体系结构对多媒体应用的软硬件支持。

图 2 为 Imagine 流处理器的硬件结构，其中有 3 个有别于通用处理器的地方：1)集成了大量的运算单元。在 Imagine 中存在 8 个运算簇(ALU cluster)，每个运算簇里面包含 3 个加法器、两个乘法器、一个除法/开方器、一个拥有 256 个入口的便签薄(Scrach-pad Memory)以及一个簇间通讯部件(Communication Unit)。48 个运算单元的计算能力对于多媒体应用中的密集计算显然是有力的支持。2)利用流寄存器文件 SRF 取代 Cache。媒体处理要求的数据供应量大，并且数据重用性极低。采用 Cache 给运算单元供应数据，不仅容量有限，无法为大量的计算单元提供数据；而且没有重用性的数据经过复杂的 Cache 机制来存取反而会降低数据访问的性能。Imagine 的 SRF 大小为 128kB，数据以流的形式存放，访问时也只能按序进行，一个流或几个流组成了数据工作集。在对当前的工作集进行处理的时候，一般都不需要进行额外的访存操作，从而将计算和访存进行了有效的分离。3)在片外 DRAM，SRF 以及本地寄存器文件 LRF 之间形成了三级带宽。运算所需数据被存放在运算簇内部的 LRF 中，保证能被频繁地快速存取；核心程序之间的中间流数据被存放在片上的流寄存器文件中，使得它能够在计算核心间循环利用而不会产生存储器访问；初始输入、最终输出和其他的全局数据存放在外部 DRAM 中，作为后备存储空间。这正好有效地利用了多媒体应用中输入数据集较大、活动工作集较小、空间局部性较强的特点，不仅保证了运算单元所需数据的及时提供，也节省了宝贵的片外带宽。

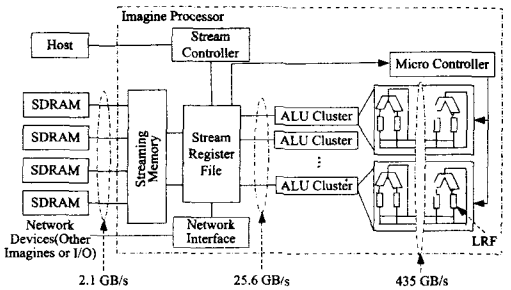

图 2 Imagine 流处理器硬件结构

为了有效地利用底层硬件，Imagine 采用两级编程模式[4]，对应了流体系结构中计算与访存的分离。流级程序主要负责数据的组织以及计算核心的划分，核心级程序则专门负责密集的计算。但由于 Imagine 存在 8 个簇，即数据是并行处理的，编程时要处理好 8 个运算簇之间的关系，特别是簇间通讯操作，要由程序员显式控制。

3 开发多媒体应用中的流并行性

3.1 多媒体基准测试程序中的流并行性

多媒体基准测试程序按其固有特点将可提取的流并行性分为以下几个方面：1)结构化的程序使得对于数据和操作的分块变得容易，这是提取流并行性的基础；2)操作块内部的数据往往要进行相同的运算操作，数据之间并没有紧密的联系，这种数据级并行是流并行的主要来源；3)前一阶段的处理结果作为后一阶段的输入，带来了计算核心间的生产者-消费者局部性，使得数据可以在计算核心之间通过 SRF 直接传递而不需要访问内存；4)大部分 video/image 测试程序的数据类型是 8bit 或者 16bit 的整型，利用流体系结构的多媒体扩展指令可以将 4 个 8bit 或者 2 个 16bit 的整型数据紧缩到 32 位的流数据中，无形中开发了数据并行。

多媒体基准测试程序中的流并行性一般存在于循环之中，这部分的计算密度最大，执行也最耗时，通过流并行加速这部分的处理才有意义。具体操作的时候，关键在于识别不存在循环依赖的代码以及可以向量化的部分，并且要具有很好的扩展性。这里所说的循环依赖主要从算法级进行考虑，即根据算法对数据的组织和处理特征来研究算法内部的数据依赖关系。理想的状态是算法中涉及的每个待处理数据执行相同的操作，并且数据之间彼此没有任何依赖。但是实际情况往往不是这样。比如视频编码中，相邻图像帧之间极为相似，为了减少需要储存的信息量，仅存储后一帧和前一帧图像间的像素差，这时相邻图像帧之间的关系就比较紧密，数据依赖较强。但因为这种依赖是数据块与数据块之间的，块内部的依赖较小，所以可开发的数据级并行仍然是可观的。还有一种情况，在语音编码中可以看到，数据依赖就存在于一帧语音的相邻信号之间，并且经常会无规则地访问某些参数，整个执行过程杂乱无章，此时的并行性就比较难提取。这种算法级的数据依赖从根本上决定了是否能够从应用中提取流并行性。另外一个影响流并行性提取难易度也是决定所提取的流并行性是否能被流体系结构充分利用的关键因素，是计算核心之间的生产者-消费者局部性，表现在多媒体应用中就是结

构化的操作块之间产生数据与消费数据的关系。这个因素决定了一个计算核心所产生的输出是否能直接被后续的计算核心使用，也部分决定了访存延迟所占的比例。

3.2 流程序开发中的关键技术

流体系结构给编程模型暴露了很多的底层细节，如数据流的组织形式、流数据在运算簇上的放置、运算簇之间的通讯等等，所以程序员要结合底层硬件结构来做具体的流程序设计。

流级程序主要是组织数据流以及划分计算核心。流是记录的有序排列，且不允许随机访问，但媒体处理中难免会存在一些不规则访存的现象，所以单一的顺序流不能满足程序设计的需求。流编程模型提供了多种可供选择的派生流，其中索引流是一种最为灵活的方式，它可以根据索引序列号对基本流进行任意顺序的访问。组织好数据流以后，理论上是把当前数据流上所做的全部操作划分到一个计算核心中，以便能充分利用数据的空间局部性。而多媒体应用具有结构化的特征，每个函数都是一个相对封闭的工作集。考虑到负载平衡以及依赖相关性，一般将一个函数拆分成几个计算核心，或者是将几个函数合并成一个计算核心。拆分后的计算核心之间往往具有生产者-消费者局部性，合并后的计算核心则具有更多的数据局部性。

核心级程序设计需要更多地考虑流数据在运算簇上的放置以及运算簇之间的通讯等。有两种设计模式可供选择：1）数据并行模式（如图 3 所示）。这是最普遍的一种设计模式，因为在绝大多数处理中，局部数据之间总是存在一定的联系，数据与操作需要作为一个整体来执行，而利用运算簇之间的通讯可以很好地处理这种相关性。这样数据并行模式一方面较好地开发了并行性，也减少了存在冗余的现象，只是会带来较大的通信开销。2）线程并行模式（如图 4 所示）。这种模式下运算簇所接收的数据流完全独立，运算簇之间也不存在通信，但是这种设计方式对于输入数据流的组织比较严格。从宏观上看是多个输入流分别输入到不同的运算簇上，但事实上只有一个输入流，且输入数据是按照依次读入多个输入流中的一个流元素的顺序来组织的。在实际的应用中，采用线程并行模式往往是在无法使用数据并行模式的情况下，即单个处理过程中所用到的数据具有太多的依赖性（比如递归的情况），大量数据相继参与运算，只为求得一个终值，此时这些数据不可分离，只能都放到同一个运算簇上执行，只是不同运算簇上得到的是不同的终值分量。

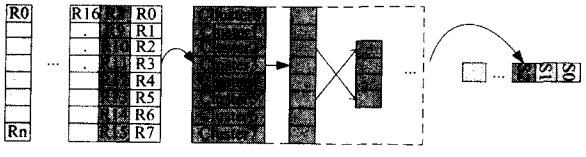

图 3 数据并行模式

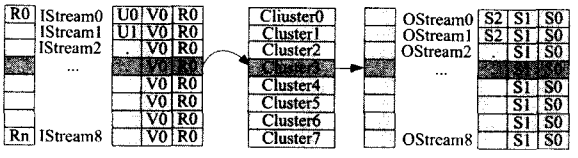

图 4 线程并行模式

4 实验分析

4.1 测评工具

我们选择了 3 个具有代表性的多媒体应用程序进行流并行程序设计以及测评，分别是图像处理程序 JPEG[5]、视频处理程序 MPEG2[6] 以及语音处理程序 Rasta[7]。流并行程序的测试平台采用斯坦福大学提供的 Imagine 流处理器的 Isim 时钟精确模拟器。串行程序的测试平台为 AMD Athlon(tm) 64 X2 Dual Core Processor 3800+，主频 2GHz，使用 linux 下的 gprof 工具收集统计信息。

4.2 结果分析

在 Imagine 上运行流程序时，整个应用的执行时间由几个部分组成：核心程序在运算簇内部的运行时间，包括核心程序花在主循环的时间、非主循环时间和运算簇暂停（等待流寄存器文件传输数据）；非核心程序运行时间，即核心程序未被启动执行其他操作的时间，存在 4 种可能：加载核心程序微代码、流控制器发射开销、标量核计算或传送流指令、等待主存数据传输。我们对 3 个应用的各种时间开销进行了统计，如图 5 所示。

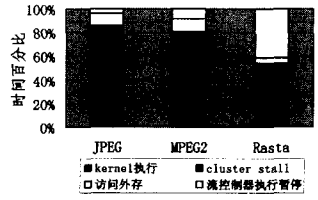

图 5 时间开销比例

从图 5 中可以看出：JPEG 和 MPEG2 大部分时间用在执行核心上。这样运算簇始终处于忙碌的状态，对计算单元的利用率比较高，说明这两个应用的核心程序中计算操作比较密集。另外这两个应用访问外存的时间相对 Rasta 要多，这是因为 JPEG 和 MPEG2 处理的数据流比较长并且数量比较多，容易产生溢出流以及双缓冲流，导致了额外的访存开销。Rasta 的情况正好与前两个应用相反，核心执行时间与流控制器暂停时间基本持平，意味着用于数据流组织调度的时间开销与真正执行计算的时间可比，说明 Rasta 的计算密集性较差，访存不规则。

我们也统计了 3 个应用执行时形成的三级带宽。从图 6 中可以看出：3 个应用的 LRF 带宽都比较高，说明对数据的访问集中在本地寄存器文件，计算核心内部的数据局部性较好；Rasta 的 SRF 带宽比较低，LRF 带宽却比较高，这是因为 Rasta 存在很多不规则的访存现象，经常要进行数据流的重组，破坏了核心间的生产者-消费者局部性，相对加重了片外 DRAM 带宽的负担；JPEG 的带宽使用情况是 3 个应用中最好的，说明 JPEG 分块合理，数据局部性和生产者-消费者局部性都相对较好。

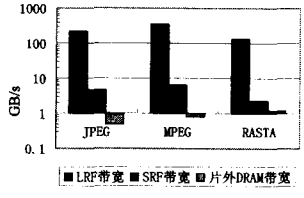

图 6 带宽层次

为了测试流并行加速的效果，我们分别统计了单个计算核心所获得的加速比以及整个应用的加速比，主要是将 Isim 模拟器上运行流程序的绝对时间与通用处理器上执行串行程序的绝对时间进行比较。单个计算核心的加速性能主要是由大量的运算部件以及 VLIW 技术所带来的，取决于核心内部的数据并行性以及指令并行性。整个应用的加速性能则从全局进行系统考虑，包括各种各样的组织调度开销，有些应用虽然单个核心的加速效果不错，但由于在组织调度上花费了太多的时间，最终的应用加速比也可能不会太好。图 7 为我们得到的统计结果。

应用程序名	主要计算核心	核心加速比	应用加速比
JPEG	color	30.5	8.6
	DCT	72.6	
	dc_ac_code	13.4	
MPEG	blocksearch_kc	23.5	5.3
	diff_kc		
	dct_kc	50.2	
	rle_kc	9.8	
Rasta	cbrand_kc	10.6	2.1
	Idft_kc	20.8	
	filt_kc		
	Powspec_kc	11.3	

图 7　多媒体应用的流并行加速性能

从图 7 中可以看出，JPEG 程序单个计算核心的加速性能较好，最终的应用加速比也相对较高，说明 JPEG 具有比较自然的数据分块特性，数据块上的计算比较密集，对数据的访存也比较规则，所以用于额外的组织调度的开销比较少，得到了比较理想的加速比；MPEG2 中由于相邻图像帧之间的依赖关系形成的反馈环在一定程度上破坏了应用的生产者-消费者局部性，而且有相当一部分计算发生在标量机上，所以虽然单个计算核心的加速比较好，整个应用加速比却只达到了 5 左右；RASTA 的性能最差，短流、计算不密集的核心使得核心加速比相对较低，而不规则的访存和控制又造成了大量的组织调度开销，所以其应用加速比也不高。

结束语　通过上述的理论分析以及实验测评，可以看出流体系结构对媒体处理做了很好的软硬件支持。利用流编程模型可以很自然地开发多媒体应用中存在的流并行性。但是一个具体的应用能否在流体系结构上取得好的流并行效果，又取决于多个方面的因素，包括多媒体应用本身具有的特点、流程序设计的方法以及使用的并行算法等等。

参 考 文 献

[1] Kapasi U J, Dally W J. The Imagine Stream Processor[C]// Proceedings of the 2002 International Conference on Computer Design. September 2002: 16-18

[2] Lee Chunho, Potkonjak M, Mangione-Smith W H. MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems[C] // Proceedings of the International Symposium on Microarchitecture. IEEE Micro-30. IEEE Computer Society Press, 1997: 330-335

[3] Fritts J, Wolf W, Liu B. Understanding multimedia application characteristics for designing programmable media processors[C] //SPIE Media Processors'99. January 1999

[4] William S A. Stream Languages and Programming Models[C]// PACT 2003. September 2003

[5] 郝杰，吴元清. 实用多媒体技术及其 C 语言实现[M]. 北京：电子工业出版社，1995

[6] 精英科技，等. 视频压缩与音频编码技术[M]. 北京：中国电力出版社，2001

[7] Lawrence Rabiner, Biing-Hwang Juang. 语音识别基本原理[M]. 北京：清华大学出版社，1999

(上接第 278 页)

不但检测出来的边缘更清晰，而且检测出原来没有检测出的一些边缘。

结束语　阈值化技术和特征空间聚类都属于并行区域分割技术，是图像分割中最重要而且有效的技术之一，在实际的图像处理系统中得到了广泛应用。特别是在需要实时性较强的图像处理系统中，快速而准确的图像阈值化方法就成为非常重要的研究目标。对一幅具体的图像，选用何种算法，要进行对比实验，不存在一种通用的图像分割算法。

参 考 文 献

[1] 秦安，冯前进，陈武凡. MR 心脏序列图像左心室内外壁联合分割和时序追踪新方法[J]. 中国图象图形学报，2008，13(1)：80-88

[2] 贾丽娟，张光年，葛庆平，等. 一种改进的彩色图像分割算法[J]. 计算机工程与应用，2008，44(13)：159-160，163

[3] Kapur J N, Sahoo P K, Wong A K C. A New Method for Gray-level Picture Thresholding Using the Entropy of the Histogram [J]. Graphical Models and Image Processing, 1985, 29: 273-285

[4] Kittler J, Illingworth J. Minimum Error Thresholding[J]. Pattern Recognition, 1986, 19: 41-47

[5] 黄伟，周鸣争，李小牛. 一种基于四元数的彩色图像边缘检测改进算法 [J]. 计算机研究与发展，2008，45(3)：121-124

[6] 徐永生. 基于小波分析的大气颗粒物数字图像的边缘检测[D]. 武汉：武汉理工大学，2006

[7] 赵海燕. 利用改进的 Prewitt 边缘算子进行车牌定位[J]. 长春理工大学学报，2005，28(1)：50-51，46

[8] Otsu N. A threshold selection method from gray-level histogram [J]. IEEE Trans. Systems Man Cybernet, 1979, 9(1): 62-66