一种确定多媒体应用程序内层循环的子字并行编译方法

吴 丹, 王志英, 王绍刚, 王 淼
(国防科学技术大学计算机学院, 湖南长沙 410073)

摘 要: 多媒体程序是数据密集型应用, 其核心代码部分占用了大部分的执行时间, 因此, 对多媒体应用程序的研究大多针对其核心代码部分. 本文围绕多媒体应用程序的核心内层循环, 分析其子字并行特点, 提出了一种新颖的按位数据流分析方法, 能够确定程序的内层循环, 为进一步深入研究奠定基础.

关键词: 多媒体; 子字并行; 内层循环; 数据流
中图分类号: TP314 文献标识码: A 文章编号: 0372-2112 (2006) 12A-2575-04

A SWP Compilation for Exploiting Inner-Loop in Multimedia Applications

WU Dan, WANG Zhi-ying, WANG Shao-gang, WANG Miao
(*Computer School, National University of Defense Technology, Changsha, Hunan 410073, China*)

Abstract: Multimedia applications are computation intensive, with kernels accounting for the majority of all dynamic execution time. Thus, research of multimedia applications mainly focuses on the code kernels. In this paper, kernel of inner loop in multimedia applications is analyzed, with its inherent parallelism: subword parallelism. And a bitwise dataflow analysis method is presented for inner loop identification.

Key words: multimedia; subword parallelism; inner loop; dataflow

1 引言

随着多媒体技术和网络的迅猛发展, 多媒体产业正在以惊人的速度扩展, 卓越的多媒体处理能力成为对现代计算机的必然要求之一.

从体系结构研究方面, 研究人员在通用处理器中增加新的多媒体功能处理部件以及相应的多媒体处理指令, 或者设计专用的多媒体处理器(ASIP), 都能够完成对多媒体信息的高效处理. 但是, 与多媒体处理器的发展相比, 面向多媒体的编译技术明显滞后. 出于多媒体应用程序与科学计算或其他通用应用程序的特殊性, 开发人员往往还是会采用复杂低效且易出错的手工编码方式产生代码.

因此, 我们认为, 无论是从计算机体系结构、还是从编译器等多方面来提高计算机的多媒体处理能力, 首先必须要针对多媒体应用特点, 对多媒体应用程序的特征进行分析和研究. 纵观历史, 有许多面向程序特征提高计算机性能的典范, 证明了对程序特征的分析和研究直接影响到计算机性能的优化和设计. 例如, IBM 公司与美国加利福尼亚大学通过对应用程序的统计和研究发现, CISC 指令集 80% 的指令只在 20% 的运行时间内用到, 因此简化了计算机指令集结构, 提出了 RISC 指令集结构, 从而提高了机器性能, 减轻设计人员负担. 循环是多媒体应用程序的核心, 在程序的运行过程中几乎占了 90% 的时间, 而核心循环体-内层循环又是循环的关键. 有实验数据表明, 大多数多媒体应用在最内层循环花费了 80% ~ 90% 的执行时间. 因此, 要挖掘多媒体应用的程序特征, 深入核心循环体的内层循环提取特征是非常有必要的. 本文围绕多媒体应用程序的特点, 对其核心代码——内层循环和并行性进行深入的分析, 针对当前编译技术的不足提出一种确定内层循环的数据流分析方法. 本文第二部分介绍并分析多媒体应用程序的核心代码, 以及其固有的子字并行性. 第三部分介绍相关编译技术的研究发展. 第四部分介绍按位的数据流分析. 第五部分给出结论.

2 多媒体应用程序内核分析

通过对多媒体应用程序定性研究分析, 多媒体应用不同于科学计算和通用应用, 它具有自身固有的特征:

(1) 内在的并行性

(2) 数据类型的连续性

(3) 实时响应性: 如视频会议和电子商务等, 需要可靠迅捷的实时响应

(4) 数据有序性: 由于弱的空间局部性, 多媒体的很多数据与操作都具有强烈的有序性

收稿日期: 2006-09-20; 修回日期: 2006-12-29
基金项目: 国家自然科学基金(No. 60173040)

(5) 高网络带宽性: 随着网络的发展, 多媒体应用对网络带宽的要求越来越高, 如网络游戏、网上视频点播等

(6) 高存储带宽性: 因为多媒体应用程序处理的数据量往往很大、甚至海量, 所以, 对存储带宽的要求也很高.

多媒体应用程序往往包含一个或多个代码核心, 代表了大部分的动态指令执行, 在程序运行过程中几乎占据了 90% 的时间, 所以, 除开以上介绍的多媒体应用程序的这些特点, 从编译的角度, 我们更关心其核心部分代码——内层循环. 一般来说, 这些核心部分执行的操作具有并行和计算密集的特点, 且代码控制结构规则, 需要对大量的、连续的、流的、可变精度的数据进行操作. 首先, 并行性的特征源于多媒体应用程序对大量独立的数据集合实现相同的操作, 因此, 这些操作能够并行执行. 第二, 大量的计算密集型的操作和代码的控制结构并不复杂, 与典型的整型程序相比, 多媒体应用程序中通常几乎不带有分支, 控制结构非常规则. 第三, 输入输出数据几乎不存在时间局部性, 因此具有流特性, 流数据即时被操作即时丢弃, 结果或者直接写回内存或者传送至网络作为输出数据流, 如流视频帧、流音频采样. 第四, 根据不同的数据类型, 数据元素的精度要求是可变的, 如 8 位的图像像素、16 位的音频采样. 一般来说, 多媒体应用程序处理的数据大小为 8 位、16 位、32 位、64 位整型, 或 32 位单精度浮点、64 位双精度浮点. 多媒体应用程序的每个数据以“子字”(subword) 的形式进行操作, 因此这种并行性称为“子字并行”(subword parallelism).

传统的通用处理器是以字为单位进行处理的, 一般是 32 位或 64 位, 而图像信息通常以大量的低精度或短数据类型的形式出现, 为了不浪费数据通路宽度, 充分高效的利用处理器上的多个计算单元, 提高处理器性能和处理多媒体信息的效率, 子字并行的处理方式应运而生. 子字并行[1] 是一种特殊的数据并行, 在子字并行中, 一个字就是一个数据集.

子字是包含在字中的更低精度的数据单元. 我们将若干个子字封装到一个字中, 然后对整个字进行处理. 子字并行允许在一个计算单元(字单元) 内对 2 个、4 个或 8 个操作(子字) 同时进行. 图 1(*a*) 所示的是一个普通的 32 位操作与相应的子字并行处理进行比较, 图 1(*b*) 为展示的是在子字单元上如何进行子字并行处理.

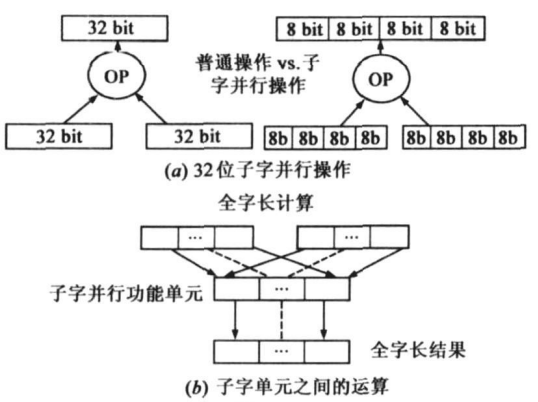

图 1

显然, 与全字长的处理相比, 子字并行操作可以带来更大的并行度. 子字并行是处理多媒体类型数据应用的最自然的方式, 它能够充分利用多媒体算法数据精度小、内部循环多的特点. 对于处理器设计来说, 子字并行大大提高了处理器的并行度, 节省了数据通路和寄存器资源; 多媒体指令的扩展又提供了更高的处理速度; 同时, 对于嵌入式多媒体处理器来说, 为支持子字并行而组织数据通路, 也可以获得更低的并行开销和更少的控制重复. 有了子字并行的处理器方式, 单一的一条 LOAD/STORE 指令就可以在存储器和寄存器之间同时移动若干个被封装的子字, 因而, 存储能力也得到了提高. 因此, 不论是通用微处理器还是多媒体微处理器或 DSP, 子字并行都是提高处理器性能、加速多媒体处理的有效方式.

3 相关研究

子字并行的处理器方式虽然很有效, 但图像处理程序主要还是以串行程序方式编写(如: C), 结果却需要以并行的方式执行, 因而对编译器提出了很高的要求, 需要编译器来决定如何将子字数据按照正确的顺序封装入源寄存器, 进行相应子字对之间的运算, 运算结束后, 按照需求, 将结果以正确的顺序从目标寄存器中提取出来. 因此, 和子字并行在体系结构研究上取得的成功相比, 通用编译器技术却不能自动的从应用程序中提取这种并行性, 进而无法充分利用这类特殊的指令集, 编译技术在自动高效的识别、产生和处理子字并行指令方面, 尚未取得令人满意的结果.

以 MMX[2~4] 为代表, 借鉴 SIMD 阵列处理技术如向量处理来产生优化代码, 采用向量处理编译技术, 能够获得 1.5 到 6.5 的加速比, 这样的结果已经相当不错. 因为对于多媒体应用这类的计算密集型应用, 内层循环是代码中最关键的部分, 也是体现最多并行的部分, 所以向量处理的目的在于识别内层循环, 采用循环变换来得到向量操作, 而这些操作就可以用子字并行指令实现. 向量处理包括: 循环分析, 循环正规化, 依赖性分析, 标量扩展, 循环分布, 向量化和循环分段.

MIT 的 Larsen 和 Amarasinghe 提出 superword[5] 的概念, 认为子字并行不仅仅是一种数据并行, 还是一种指令级并行的方式, 因此, 对于一个基本块中按相同顺序执行的相同的操作, 都可以进行并行处理. 这种方法与传统的向量编译方法的不同在于它的向量化是对一个基本块内部指令的向量化. 该方法基于 SUIF[5] 实现, 并应用于 Motorola 的 AltiVec 扩展. 这种方法解决了基本块内部的并行处理问题, 但是只适用于基本块结构简单的情况, 对于带条件转移的基本块如何处理, 并没有进行深入研究, 而且这种方法由于过于复杂, 不具备很好的移植性, 也难于实现, 所以, 只能作为一种启发式方法进行研究.

与现有研究的不同在于, 本文沿袭向量编译的原理, 根据图像处理程序的特点, 着重对关键部分——内层循环进行挖掘, 提出了按位的数据流分析方法, 为进一步编译优化技术的使用奠定基础.

4 按位数据流分析

数据流分析能够帮助我们确定内层循环, 我们首先用循

环正规化技术(loop normalization)来保证循环的迭代空间的规则性和依赖性验证的简单. 本文主要考察针对 for 循环的匹配, 在开始进行子字并行规范之前, 我们假设该循环中所有上下界都明确定义.

对于 C 编译器而言, 我们假设其规约变量下界为0、变化步长为 1、且所有循环迭代都可以运行, 则旧的规约变量将为新规约变量的仿射函数代替, 下标表达式和下界也因此做相应修改. 下例所示为一个小的循环代码段, 其规约变量下界为2、步长为 1.

```
short int a[1000], b[1000], c[1000];
for( i= 2; i< 1000; i++ )
a[ i]= b[i] + c[i];
```

通过循环正规化技术处理后, 该循环可以变形为:

```
short int a[1000], b[1000], c[1000];
for( i= 1; i< 999; i++ )
a[ i+ 1]= b[i+ 1] + c[i+ 1];
```

本文使用"格"作为数据流分析中对内部数据结构的形式化表示方法. 传统的数据流分析[6]可以看作是沿着变量位宽的传播进行操作, 或者为每个变量维持一个位向量. 前者无法得到算术操作的精确的结果, 因为对一个 8 位数的位传播往往导致一个 9 位的结果, 尽管 8 位数就足够了, 或者一个变量中只有最重要的几个位能够进行位消去. 而后者不支持精确的算术分析, 因为它往往会保守的假设每个加法结果都产生进位. 所以, 在进行数据流分析的时候, 我们使用"格"结构来形式化表达内部数据结构. 一个数据范围可以看作是从某个下界到某个上界变化的整数的子范围的简单连接, 因此, 用数据范围就能够清楚的表达某个变量的下界和上界. 因为我们只需要用一个简单的范围就可以表示变量的所有可能的值, 因此, 尽管该表示方法不允许消去低位的位, 却可以对算术表达式实现精确的计算.

我们选择格来表示数据范围传播基于以下三个方面的因素: 首先, 这种表示方法能够将位宽分析应用到更普遍的值范围传播问题, 这对于解决值预测、分支预测、常数传播、过程复制和程序验证都非常有效. 第二, 该表示方法保证了精确性, 这对于大多数算术应用非常重要; 第三也是最重要的, 在我们的编译算法中, 用生存期(life range)作为处理子字的标准, 在该结构中, 我们将每个子字的生存期映射到值的范围, 从而获得每个生存期的精确结果, 所以, 这种"格"的结构也可视为对生存期传播的表示.

在格的表示中, 值的生存期被分配到某个变量, 格自下向上提升, 如图 2 所示值的定义和计算如下:

(1) $\perp_{DR_\perp}$: 未初始化时的子字生存期的值

(2) $T_{DR_\perp}$: 不能静态决定的值, 即子字集合的生存期的上界

(3) $\cup$: 生存期的并集 $\langle a_l, a_h\rangle \cup \langle b_l, b_h\rangle = \langle \min(a_l, b_l), \max(a_h, b_h)\rangle$

(4) $\cap$: 生存期的交集 $\langle a_l, a_h\rangle \cap \langle b_l, b_h\rangle = \langle \max(a_l, b_l), \max(a_h, b_h)\rangle$

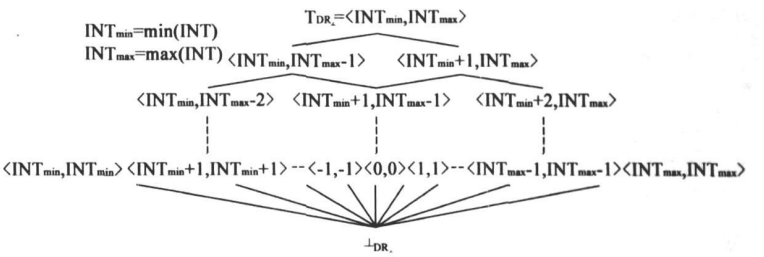

图 2　为变量分配子字值的格表示

以如下所示的代码段为例, 这是取自 MPEG-4 算法中的一段简化的循环代码,

```
for  ( i= 0; i< h; i++ ) {
C[ i]= A[ i]- B[ i] /* 1 S* /
D[ i+ 1]= D[ i]+ C[ i] /* 2 S* /
}
```

我们通过分析, 得到代码段中的真相关和输出相关, 图 3 分别表示循环的数据相关以及循环分配后得到的强连通部件.

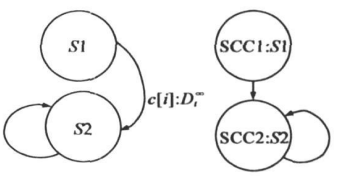

图 3　数据相关图及其强连通部件

相关图的强连通部件表示带有向边的节点的最大集合, 带有多个节点的强连通部件表示相关循环中语句的最大集合.

语句 $S1$ 不带自圈, 所以它自己组成了一个单一的强连通部件, $S1$ 能够按照子字并行方式处理. 语句 $S2$ 在循环中是自相关的, 我们用循环展开来消除这种相关, 如下代码所示.

```
for  ( i= 0; i< h; i+ = 2) {
C[ i]= A[ i]- B[ i];
C[ i+ 1]= A[ i+ 1]- B[ i+ 1]}
```

首先根据操作数的类型进行适当的展开, 如寄存器为 32 位, 那么对于 C 语言中定义的 short 类型的 int 操作数, 就要将循环展开 2 次, 然后进行非循环指令调度, 组合相同循环指令的所有实例, 这样, $S2$ 也可以按照子字并行方式执行了.

5　结论与展望

本文根据多媒体应用程序的特点, 对其关键核心——内层循环进行分析研究, 在现有向量编译的基础上, 提出了挖掘内层循环的按位数据流分析算法, 方法简单, 容易执行, 很好的挖掘了图像处理程序内在固有的子字并行. 我们将在此工作基础上, 进一步融合多媒体编译优化技术, 在更复杂的循环结构上进行分析研究.

参考文献:

[1] R B Lee. Subword parallelism with max-2[J]. IEEE Micro, 1996, 16(4): 51- 59.

[2] A Krall, S Lelait. Compilation techniques for multimedia processors[J]. International Journal of Parallel Programming, 2000, 28(4): 347- 361.

[3] Intel Corporation. IA-32 Intel Architecture Software Developer's Manual[Z]. Volume 2A: Instruction Set Reference, A-M. 2004.

[4] Intel Corporation. IA-32 Intel Architecture Software Developer's Manual[Z]. Volume 2B: Instruction Set Reference, N-Z. 2004.

[5] Samuel Larsen, Saman Amarasinghe. Exploiting superword level parallelism with multimedia instruction sets[J]. ACM SIGPLAN Notices, 2000, 35(5): 145–156.

[6] Randy Allen, Ken Kennedy. Optimizing Compilers for Modern Architectures: A Dependence-Based Approach[M]. San Mateo, USA: Morgan Kaufmann, 2002.

作者简介:

吴 丹 女, 1979 年 10 月出生于湖南祁东, 国防科学技术大学计算机学院博士, 主要研究方向为先进编译技术、高级计算机体系结构. E-mail: daisydanwu@nudt.edu.cn

王志英 男, 1956 年 8 月出生于山西长治, 国防科学技术大学计算机学院博士, 教授, 博士生导师, 主要研究方向为高级计算机体系结构. E-mail: zywang@nudt.edu.cn