软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2011,22(7):1475−1487 [doi: 10.3724/SP.J.1001.2011.03847]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

异构多处理器 SoC 的应用算法性能优化方法*

赵 鹏+, 严 明, 李思昆

(国防科学技术大学 计算机学院,湖南 长沙 410073)

Performance Optimization of Application Algorithms for Heterogeneous Multi-Processor System-on-Chips

ZHAO Peng+, YAN Ming, LI Si-Kun

(School of Computer, National University of Defense Technology, Changsha 410073, China)

+ Corresponding author: E-mail: pengzhao@nudt.edu.cn

Zhao P, Yan M, Li SK. Performance optimization of application algorithms for heterogeneous multi-processor system-on-chips. *Journal of Software*, 2011,22(7):1475−1487. http://www.jos.org.cn/1000-9825/3847.htm

Abstract: MPSoCs (multi-processor system-on-chips) are comprehensively applied in the embedded multimedia processing field. Multimedia MPSoCs often adopt the "host processor+multiple heterogeneous synergistic processor" architecture, which makes the trade-off between universality and flexibility. MPSoCs also take into consideration both performance and power-consumption, but challenges the method of performance optimization of System-on-Chip applications. This paper proposes an approach that improves the performance of application algorithms running on heterogeneous MPSoCs. The approach includes three stages: Application feature analysis, affine partitioning of kernel loops, and "application-architecture" mapping. It optimizes the multi-level parallelism and data locality of application algorithms to improve MPSoC performance. Experimental results show that the proposed approach can greatly improve the multimedia processing performance on heterogeneous MPSoCs.
Key words: system-on-chip; multi-processor system-on-chip; embedded system; multimedia processing

摘 要: 在嵌入式多媒体处理领域中,多处理器片上系统(multi-processor system-on-chip,简称 MPSoC)的应用越来越广泛.多媒体处理 MPSoC 通常采用"主处理器核+多个异构协处理器核"的主流体系结构.该结构兼顾了 MPSoC 系统的通用性与灵活性、性能与功耗,但也向 MPSoC 的性能优化方法提出了更高的要求.针对异构 MPSoC 上的多媒体应用算法,提出了一种 MPSoC 多媒体处理性能优化方法.该方法经过应用特征分析、循环仿射划分、应用向 MPSoC 各处理器核的映射,实现了优化的数据局部性与多级并行性,从而提高了异构 MPSoC 上多媒体应用算法的性能.实验结果表明,该方法对于多媒体应用算法在异构 MPSoC 上的处理性能优化方面取得了明显效果.
关键词: 片上系统;多处理器片上系统;嵌入式系统;多媒体处理
中图法分类号: TP311 **文献标识码**: A

在嵌入式多媒体信息处理领域,SoC(system-on-chip)技术的应用非常广泛.随着集成电路技术的发展与用

* 基金项目: 国家自然科学基金(90207019, 90707003)
收稿时间: 2009-06-30; 定稿时间: 2010-03-15

© 中国科学院软件研究所 http://www.jos.org.cn

户需求的不断提高,出现了多处理器 SoC(multi-processor system-on-chip,简称 MPSoC)体系结构.MPSoC 通常在单颗芯片上集成一个通用的嵌入式 RISC 处理器核和多个协处理器核以及配套的内部存储器和输入/输出等设备.采用异构 MPSoC 体系结构,通过增加处理器核的种类与数量,兼顾了系统的通用性与灵活性、性能与功耗,并降低了系统成本.然而,该结构的复杂性对 MPSoC 的应用性能优化提出了更高的要求,对异构 MPSoC 上应用算法的性能优化工具有着更迫切的需求.

在多媒体处理 MPSoC 的已有研究中,典型代表包括 ST Nomadik[1],Viper[2],VISoC[3],ESP-MPSoC[4]等.这些异构 MPSoC 平台都采用了“主处理器核+多个异构协处理器核”的体系结构.在上述异构 MPSoC 中,利用主处理器核进行系统控制等非计算密集型的运算,而利用协处理器核进行多媒体处理中的计算密集部分.异构 MPSoC 上应用算法的实现,依赖于程序员的编程工作与人工优化.应用处理时,MPSoC 内各处理器核的分工借助性能分析工具和设计经验来完成.

上述研究中,过于依赖程序员及其设计经验进行异构 MPSoC 上应用算法的性能优化,不能应对 MPSoC 应用越来越高的复杂性,会导致开发效率低、不能充分发挥异构 MPSoC 计算能力、性能优化机会大量损失等问题.特别是随着集成电路技术的发展和用户对多媒体处理质量与效率的需求日益提高,异构 MPSoC 功能和结构的复杂度必然快速增大[5].在这种趋势下,上述问题会更加严重.

在多处理器 SoC 应用优化的已有研究中,PluTo[6]和 LeTSeE[7]是具有代表性的自动优化框架.这两个框架使用了多面体模型和整数线性规划方法进行程序并行转换,转换后的程序具有良好的通信开销和数据局部性.但对于异构 MPSoC,这些通用优化框架没有考虑异构 MPSoC 体系结构特征(存储层次特征与 MPSoC 各处理器核的差异)和“应用功能-MPSoC 各处理器核”的映射策略,导致体系结构相关的优化不充分和负载不平衡,在异构 MPSoC 上应用时还需要进行扩充与改进.

根据目前方法的不足及未来发展的趋势,本文针对异构 MPSoC 上的多媒体处理应用提出一种性能优化方法,通过并行性与数据局部性的优化,提高异构 MPSoC 的处理性能.该方法首先对异构 MPSoC 上的应用进行特征分析,根据分析结果,将核心循环划分为可并行执行的多级循环子块.多级循环子块在各级存储层次上具有优化的数据局部性,使用组合寻优算法将循环子块映射到 MPSoC 各处理器核上执行,从而实现了循环子块和异构处理器核之间的多级并行.在实例研究中,使用一种新型的异构 MPSoC 和典型的多媒体处理算法进行了实验.实验结果表明,该方法明显地提高了异构 MPSoC 上的多媒体处理性能.

本文第 1 节给出使用的基本术语、假设和定理.第 2 节定义异构 MPSoC 上应用算法的性能优化问题.第 3 节阐述异构 MPSoC 上应用算法的性能优化方法.第 4 节给出实验结果与分析.第 5 节总结本文,并对未来工作进行展望.

1 基本术语、假设与定理

使用如下定义来表示异构 MPSoC 上的应用程序,后面的描述与表示都基于定义 1 中所定义的符号.

定义 1(程序的表示)[8]. 异构 MPSoC 上的串行程序可以表示为 $P:\langle I,\delta,\Psi,\Upsilon,\omega,\eta\rangle$.其中:

- I 为指令集合,每条指令是一个不可分的单元.设 $s\in I,t\in I$,当且仅当 $s<_p t$,指令 s 先于指令 t 执行;
- δ_s 表示包围指令 s 的循环嵌套深度;
- $\Psi_s(\boldsymbol{i}_s)=\boldsymbol{D}_s\boldsymbol{i}_s+\boldsymbol{d}_s$ 为循环边界的仿射表达式.其中,$\boldsymbol{i}_s=(i_1,i_2,\ldots,i_n)^T$ 为指令 s 的循环索引.$(i_1,i_2,\ldots,i_n)$从左至右分别代表最外层循环索引变量直至最内层循环索引变量.当且仅当 $\Psi_s(\boldsymbol{i}_s)\geqslant 0$ 时,$\boldsymbol{i}_s$ 有效.$\Psi_s(\boldsymbol{i}_s)$定义了循环的迭代空间 L^n;
- $\Upsilon_{zsr}(\boldsymbol{i}_s)=\boldsymbol{H}_{zsr}\boldsymbol{i}_s+\boldsymbol{f}_{zsr}$ 为指令 s 中数组 z 第 r 个引用的数组索引的仿射表达式;
- ω_{zsr} 为真,当且仅当指令 s 中数组 z 第 r 个引用为写操作;
- η_{st} 为指令 s 和指令 t 共同嵌套的循环层数.

在异构 MPSoC 的应用中,多媒体处理属于计算密集型应用,需要进行计算加速.在多媒体处理中,循环嵌套与多维数组操作占据了大部分执行时间.因此,将性能优化的重点放在循环的并行与数据局部性优化上,并给出

© 中国科学院软件研究所 http://www.jos.org.cn

下列假设:(1) 循环内以仿射形式进行数据访问;(2) 程序中嵌套循环只包含数组变量,对于标量,可以通过标量扩张或标量私有化对其进行预处理;(3) 对嵌套循环进行了归一化(normalized)处理,使每层循环的索引变量都从 0 开始,且以 1 为步长递增变化[9].

异构 MPSoC 上应用算法的性能优化分为 3 个步骤:应用特征分析、循环仿射划分和应用功能到 MPSoC 各处理器核的映射.其中,循环仿射划分需要保证程序转换前后的语义一致性,因此需要满足一些约束条件.文献[8]针对不同级别的同步需求,分别给出了循环仿射划分前后的一致性约束.这里,基于该文献推导出循环仿射划分有效的必要条件.

定理 1(循环仿射划分有效的必要条件). 设程序 P,指令 $s,t\in I$,且 $s\neq t$.根据 P 中数据 z 的相关性进行仿射划分 ϕ.ϕ 有效的必要条件为:$\forall\langle \Upsilon_{zsr},\Upsilon_{ztr'}\rangle$,对于满足条件$(\Psi(\boldsymbol{i}_s)\geqslant 0)\wedge(\Psi(\boldsymbol{i}_t)\geqslant 0)\wedge(\Upsilon_{zsr}(\boldsymbol{i}_s)=\Upsilon_{ztr'}(\boldsymbol{i}_t))$的 $\boldsymbol{i}_s\in\Upsilon^{\delta_s}$和$\boldsymbol{i}_t\in\Upsilon^{\delta_t}$,有$\phi(\boldsymbol{i}_t)-\phi(\boldsymbol{i}_s)\geqslant 0$.

证明:根据数据 z 相关性的仿射划分,ϕ 将指令划分为不同的迭代集合,并且要保证各个迭代顺序执行时不能违反原有的数据相关性.条件$(\Psi(\boldsymbol{i}_s)\geqslant 0)\wedge(\Psi(\boldsymbol{i}_t)\geqslant 0)\wedge(\Upsilon_{zsr}(\boldsymbol{i}_s)=\Upsilon_{ztr'}(\boldsymbol{i}_t))$的值对$\langle \Upsilon_{zsr},\Upsilon_{ztr'}\rangle$指定了 s 和 t 之间关于数据 z 的相关性.这些数据相关性在仿射划分之前为 $s<_p t$,则要保证 $\boldsymbol{i}_s\leqslant\boldsymbol{i}_t$.而仿射划分之后,指令 s 和 t 对应的迭代向量为$\phi(\boldsymbol{i}_s)$和$\phi(\boldsymbol{i}_t)$,仍要满足数据相关性:指令 t 要么与指令 s 在同一迭代执行$\phi(\boldsymbol{i}_s)=\phi(\boldsymbol{i}_t)$,要么在指令 s 迭代执行之后执行$\phi(\boldsymbol{i}_s)<\phi(\boldsymbol{i}_t)$,即$\phi(\boldsymbol{i}_t)-\phi(\boldsymbol{i}_s)\geqslant 0$. □

引理 1(Farkas 引理)[10]. 设 D 为非空多面体,由 s 个仿射不等式定义:$a_k x+b_k\geqslant 0$,其中,$1\leqslant k\leqslant s$,则 D 中每个仿射$\phi(x)$都是非负的,当且仅当其为不等式所定义表面的正仿射组合,即

$$\phi(x)\equiv\lambda_0+\sum_k\lambda_k(a_k x+b_k),$$

其中,$\lambda_k\geqslant 0$ 为 Farkas 乘子.

定理 2[6]. 假定函数$\delta_e=\phi(\boldsymbol{i}_t)-\phi(\boldsymbol{i}_s)$.其中,满足条件$(\Psi(\boldsymbol{i}_s)\geqslant 0)\wedge(\Psi(\boldsymbol{i}_t)\geqslant 0)\wedge(\Upsilon_{zsr}(\boldsymbol{i}_s)=\Upsilon_{ztr'}(\boldsymbol{i}_t))$的$\langle \Upsilon_{zsr},\Upsilon_{ztr'}\rangle$构成集合 E. $\boldsymbol{i}_s\in\Upsilon^{\delta_s}$和$\boldsymbol{i}_t\in\Upsilon^{\delta_t}$. 如果所有迭代空间有界,则存在一个仿射函数 $v(\boldsymbol{p})=u\boldsymbol{p}+w$,该函数可以限制$\delta_e$ 的上界.即,$\forall e\in E, v(\boldsymbol{p})-\delta_e\geqslant 0$.其中,$\boldsymbol{u}=(u_1,u_2,\ldots,u_k)$,$p$ 为循环嵌套的结构参数向量.

2 问题定义

2.1 异构MPSoC的功能模型

为了进行应用特征分析与优化,需要建立异构 MPSoC 的功能模型.异构 MPSoC 的功能模型采用 PIA-CFTG (program information aided control flow task graph)[11].PIA-CFTG 模型在任务图的基础上增加了应用特征属性(如控制流关系、执行时间、核心循环).PIA-CFTG 模型通过程序分析与系统建模技术得到[11].该模型定义如下:

定义 2(PIA-CFTG 模型). PIA-CFTG 模型定义为有向图 $G_f=(T,E,P)$.其中:

- T 是节点的集合.节点 T 可以表示 3 种粒度:基本块、循环和函数.每个节点 t 对应一个节点属性值 $P_t\in P$.在节点中,t_0 和 t_n 为特殊节点,分别表示程序的开始和结束.开始节点没有前驱,结束节点没有后继.程序执行的控制流从开始节点进入,从开始节点可到达图中任意一个节点,程序的控制流由结束节点流出;
- $E=\{e_{ij}|t_i>t_j,t_i\in T,t_j\in T\}$是有向边的集合.$t_i>t_j$ 表示从 t_i 到 t_j 的有向边,也表示出节点 t_i 和 t_j 间的控制流关系;
- P 表示所有节点的属性集合 $P=\{P_t|0\leqslant t\leqslant N-1\}$.其中,$N$ 为节点的数目,属性值包括执行时间、存储需求量、数据活跃性、数据相关性、计算/操作数特征、核心循环特征、节点重要性因子等.

当 PIA-CFTG 模型的任务节点为循环时,该模型表示应用程序中的循环信息;当 PIA-CFTG 模型的任务节点为仿射划分生成的循环子块(见第 3.2 节)时,该模型表示应用程序中的循环子块信息.

2.2 异构MPSoC的体系结构模型

使用 AG(architecture graph)模型[12]描述异构 MPSoC 的体系结构,AG 模型是 MPSoC 应用所运行平台的抽象,用于描述体系结构信息以支持体系结构相关的性能优化.AG 模型定义如下:

© 中国科学院软件研究所 http://www.jos.org.cn

定义 3(体系结构图 AG). 体系结构图定义为 G_a={V,E},其中:

- V 为顶点集合,每个顶点表示一个体系结构部件.体系结构部件分为两类:处理器部件 P:{p_1,p_2,…,p_r}和存储器部件 M:{m_1,m_2,…,m_s};
- E 为边的集合,表示体系结构部件之间的连接;
- $\forall p\in P$,集合 $M_p:\{m_{p_1},m_{p_2},...,m_{p_t}\}$ 表示处理器 P 可以访问的存储器集合.

以图 1 中的嵌入式流媒体处理 SoC(embedded stream processing multi-processor SoC,简称 ESP-MPSoC)[4]为例,描述异构 MPSoC 体系结构及其 AG 模型.ESP-MPSoC 由课题组自主研发并投片成功.ESP-MPSoC 集成了 1 个 32 位 RISC 通用嵌入式微处理器核(EStarIII)和 2 个专用协处理器核(*SP*1 和 *SP*2).其中,协处理器核 *SP*1 和 *SP*2 均采用 128 位的 SIMD 数据通路,支持最大 16 个字节数据的并行操作,从而良好地支持嵌入式多媒体应用.*SP*1 和 *SP*2 采用一个共同的 32 位编码的 SIMD 基本指令集.因此,*SP*1 和 *SP*2 的流水线比较相似.不同的地方在于:针对应用定制的特殊指令和特殊功能单元 SFU,*SP*1 扩展了一些特殊指令,如绝对差值和(SAD)、寄存器数据重排序(RRD)、寄存器数据选择(RDS)等;*SP*2 则增强了正弦、余弦、求对数、求指数、定点 ALU 指令和饱和运算等.*SP*1 和 *SP*2 的片上存储器采用分块设计,即每片存储器分为两个独立访问的部分来设计.片上存储器与协处理内核之间采用 128 位的大位宽访存设计,而对通信控制单元,则提供 64 位存储读写端口,从而极大地提高了各个模块之间的通信带宽.

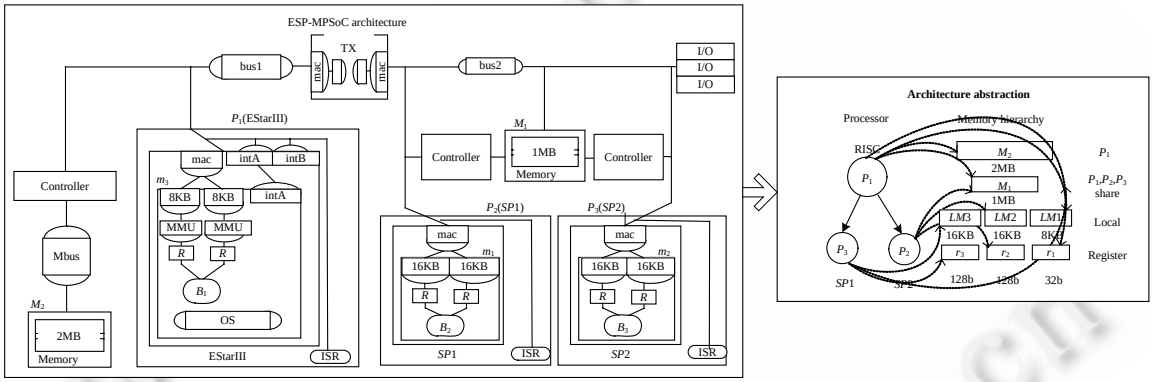

Fig.1 ESP-MPSoC architecture and AG model

图 1 ESP-MPSoC 体系结构及其 AG 模型

2.3 异构MPSoC上应用算法的性能优化问题

异构 MPSoC 的应用算法性能优化方法分为 3 个步骤:应用特征分析、循环仿射划分和应用功能到 MPSoC 各处理器核映射.其中:通过应用特征分析得到 MPSoC 应用的特征,如控制流信息、相关性、执行时间和关键循环等;循环仿射划分将核心循环划分为多级循环子块,以支持循环子块级的并行,并且多级循环子块在对应的各层存储层次上具有优化的数据局部性;映射是指使用组合寻优算法将循环子块映射到 MPSoC 各处理器核上,以支持 MPSoC 各处理器核间的并行与负载平衡.下面分别定义仿射划分与映射问题.

定义 4(循环仿射划分子问题). 针对程序 P 中的单条指令 $s\in I$ 的仿射划分为 $\phi_i(\boldsymbol{i}_s)=\boldsymbol{C}\boldsymbol{i}_s+\boldsymbol{c}_s$.其中,$1\leqslant i\leqslant k$,$k$ 为 P 中的指令条数.针对程序 P 的仿射划分为 $\phi=[\phi_1,\phi_2,\ldots,\phi_k]$.仿射划分后,原有循环迭代空间被划分成循环子块.这些循环子块由超平面簇 H:{H_0,H_1,…,H_{n-1}}构成.仿射划分问题即为:在满足约束条件的情况下,寻找 P 的仿射划分 ϕ,以实现循环迭代的重新组合与排列,并获得最优目标函数值.

循环仿射划分问题中的目标函数和约束条件定义如下:

- 目标函数 $OF_1=\phi(\boldsymbol{i}_t)-\phi(\boldsymbol{i}_s)$,即循环仿射划分后的程序 P'中,相关数据的相关距离最短.
  其中,$\boldsymbol{i}_s\in\Upsilon^{\delta_s}$,$\boldsymbol{i}_t\in\Upsilon^{\delta_t}$,且满足条件($\Psi(\boldsymbol{i}_s)\geqslant 0)\wedge(\Psi(\boldsymbol{i}_t)\geqslant 0)\wedge(\Upsilon_{zsr}(\boldsymbol{i}_s)=\Upsilon_{ztr'}(\boldsymbol{i}_t))$;

© 中国科学院软件研究所 http://www.jos.org.cn

- 约束条件:

(1) 程序仿射划分有效,即满足定理 1 给出的必要条件;

(2) 满足相关源和相关目标的索引间的仿射关系 $f:\boldsymbol{i}_s\rightarrow\boldsymbol{i}_t$;

(3) 程序仿射划分后,索引变量仍需在迭代空间内,即,$\phi(\boldsymbol{i}_s)\in L^n$ 且 $\phi(\boldsymbol{i}_t)\in L^n$.

定义 5(映射子问题-图节点多着色). 将 PIA-CFTG 图 G_f 的节点设置为循环子块,则循环子块向 MPSoC 各处理器核的映射,可以转换为 G_f 节点的多着色问题.设有 n 个循环子块向 m 个处理器核上映射,则 G_f 节点数目为 n+2 个(循环子块、开始节点和结束节点),每个节点可选的颜色有 m 种.使用函数 $color(t_i,c_k)$=1,表示将 G_f 节点 t_i 着色为颜色 c_k,着色方案用着色向量 $\boldsymbol{v}$=[$c_0,c_1,\ldots,c_{n-1}$]来表示.其中,$c_k\in\{0,1,\ldots,m-1\}$,$k\in[0,n-1]$.循环子块向 MPSoC 各处理器核的映射,即寻找 G_f 节点的一个着色方案 $\boldsymbol{v}$,使得在满足约束条件的情况下目标函数值最小.

在图节点多着色问题中,目标函数与约束条件如下:

- 目标函数 $OF_2=\max\left\{\sum_{i=1}^{k_1}time(N_i^1),\sum_{i=1}^{k_2}time(N_i^2),\ldots,\sum_{i=1}^{k_m}time(N_i^m)\right\}$.即:将循环子块向 MPSoC 各处理器核映射之后,应用处理时 m 个处理器核中最长的处理时间.对于 MPSoC 的第 j 个处理器核,其执行时间为 $\sum_{i=1}^{k_j}time(N_i^j)$. 其中,$1\leqslant j\leqslant m$,$k_j$ 为已映射到处理器核 j 上的节点数目,$time(N_i^j)=C(N_i^j)+D(N_i^j)$. $C(N_i^j)$ 为计算时间,通过 ModelSim 模拟得到. $D(N_i^j)$ 为数据准备与回传时间;
- 约束条件:图节点多着色过程需要同时满足下列两个约束条件:

(1) $\forall i,j\in[1,n-1]$,若 $i=j$,$color(t_i,c_i)=1$,$color(t_j,c_j)=1$,则 $c_i=c_j$.即 G_f 其中任何一个节点,只能具有一种颜色;

(2) $\forall i\in[1,n-1],\bigcup_{k=0}^{m-1}color(t_i,c_k)=1$.即 G_f 中除了开始节点和结束节点之外,任何节点都必须着色.

3 异构 MPSoC 的应用算法性能优化方法

3.1 异构MPSoC的应用特征分析

异构 MPSoC 的应用特征分析,采用动态和静态分析相结合的方法.

静态分析的主要任务为:

(1) 基于 SUIF 编译器框架的控制流库进行控制流分析,并逐步构造 PIA-CFTG 模型;

(2) 基于 LooPo 工具进行相关性分析,并构造描述多面体相关性的多面体相关图(polyhedral dependence graph,简称 PDG).PDG 定义如下:

定义 6(多面体相关图 PDG). PDG 是一个有向多图,用多元组〈V,E,M,R,T〉来表示.PDG 中所有的集合和关系都是凸多面体的整数向量集合.其中:

- V 是顶点的集合.每个顶点表示一个数据相关性对应的操作集合,例如循环中同一条指令的所有迭代操作.对于每个顶点 $v\in V$,用多面体 M_v 表示该顶点对应操作集合的域.每个顶点的一次操作可以表示为一个值对(v,x).其中,x 为整数向量且 $x\in M_v$;
- E 是顶点间的边的集合.每条边 $e\in E$ 表示其起点操作 $\boldsymbol{i}_s$ 与终点操作 $\boldsymbol{i}_t$ 间关于数据 z 的相关性.$\forall e\in E,e$ 对应两个多面体(R_{es} 和 R_{et})和一个仿射划分矩阵 T_e.R_{es} 和 R_{et} 分别表示相关源和相关目标的域;T_e 表示相关源到相关目标的仿射关系,即 $f:\boldsymbol{i}_s\rightarrow\boldsymbol{i}_t$.

动态分析的主要任务为:

(1) 分析异构 MPSoC 上应用程序的各个部分(如核心函数、核心循环、循环子块等)在不同处理器核上的执行时间;

(2) 通过“插桩”方法,分析异构 MPSoC 上应用处理的过程中,各个数据影响应用处理性能的权重值.

© 中国科学院软件研究所 http://www.jos.org.cn

3.2 基于关注元的多级仿射划分

定义 7(关注元与非关注元). 关注元是指程序中对仿射划分产生影响的数组,非关注元是指程序中对仿射划分不产生影响的数组.在不同级别的仿射划分中,关注元可以不同.当前级别的仿射划分中使用的关注元,称为当前关注元.关注元的选择标准为:该数组影响程序执行性能的权重.

在仿射划分过程中,根据关注元的相关性将循环的迭代空间划分为可并行的循环子块.多级仿射划分具有以下 3 个主要性质:

(1) 多级仿射划分与各级存储层次相对应,使划分结果在各级存储层次(如寄存器、局部存储器、共享存储器等)上具有优化的数据局部性;

(2) 各循环子块执行期间,循环子块间无需关于该数据进行通信;

(3) 单个循环子块可在各处理器核上单独运行,不同处理器核可并行处理各个循环子块的计算和数据传输.

下面阐述基于关注元的多级仿射划分过程.

步骤 1. 确定仿射划分的参数.读取体系结构图模型 G_a 中的存储层次信息,推导出仿射划分的参数:级数和循环子块的大小.

读取异构 MPSoC 的 G_a,得到其存储层次信息,推导出循环多级仿射划分的级数和循环子块的大小.假设 MPSoC 为 3 级存储层次,其中,L_1,L_2 和 L_3 级的存储器容量分别为 M_1,M_2 和 M_3,则 L_1,L_2 和 L_3 级循环子块的大小分别为 $sl\_size_1=\alpha M_1$,$sl\_size_2=\beta M_2$ 和 $sl\_size_3=\min\left(\frac{M_3}{ceil(M_3/\alpha M_1\beta M_2)},\gamma M_3\right)$.其中,$\alpha,\beta,\gamma\in(0,1)$.通过调整各级循环子块的大小,使其数据尽量能够同时保存在对应的存储层次中,从而提高数据局部性,减少存储层次之间的数据传输.

循环仿射划分的级数 $n=\begin{cases}2, \text{ if } sl\_size_2 \geqslant \lambda_1 sl\_size_3\\ 3, \text{ if } sl\_size_2 < \lambda_1 sl\_size_3\\ 3, \text{ if } sl\_size_3 \geqslant MEM\_SIZE\end{cases}$.其中,系数 $\lambda_1,\lambda_2\in(0,1)$.

步骤 2. 关注元的选取与相关性分析.根据程序中操作的各个数据在程序执行时间性能中的权重,选取权重值最大的数据作为当前关注元 z,并对其进行相关性分析(方法见第 3.1 节),得到多面体相关图模型 PDG^z.

步骤 3. 多级仿射转换.使用整数线性规划,寻找一个使目标函数值最小的多级仿射转换.在仿射划分的参数和当前关注元 z 确定之后,采用文献[6]中的方法,针对当前关注元 z 进行单级仿射划分.首先,寻找构成循环子块的第 1 个超平面簇,该超平面簇包含关注元 z 的最大相关距离.使用目标函数 OF_1,寻求使 OF_1 最小的仿射划分.将引理 1 和定理 2 联合,得出:

$$v(\boldsymbol{p})-OF_1=(u\boldsymbol{p}+w)-(\phi^z(\boldsymbol{i}_t)-\phi^z(\boldsymbol{i}_s))=\lambda_{e0}+\sum_k\lambda_{ek}\left(c_{ek}\left(\frac{\boldsymbol{i}_s}{\boldsymbol{p}}\right)+d_{ek}\right)\geqslant 0.$$

将 $\boldsymbol{i}_t$ 进行 h 转换[8]成 $\boldsymbol{i}_s$ 后,上式转换为 $f(u,w)=g(\boldsymbol{i}_s)$)的形式,且可由 $\boldsymbol{A}\boldsymbol{i}_s+\boldsymbol{b}\geqslant 0$ 推导出 u 和 w 的约束条件.对关注元 z 的每一个相关性都求出一组约束条件,并组成约束集合.在满足所有约束的前提下,寻找 u 和 w 的最小字典序 $\min_{\prec}\{u_1,u_2,...,u_k,w,...,c_i's,...\}$.该最小字典序决定了一个超平面,且该超平面在法向量方向没有相关性的超平面.该超平面即为寻找的第 1 个超平面簇.

基于第 1 个超平面簇,迭代求解构成循环子块的其他超平面簇.

设矩阵 $\boldsymbol{H_s}$ 的行表示已知超平面,待求解的超平面必须与已知超平面相互正交,即在 $\boldsymbol{H_s}$ 的正交子空间 $\boldsymbol{H}_s^{\perp}=\boldsymbol{I}-\boldsymbol{H}_s^T(\boldsymbol{H}_s\boldsymbol{H}_s^T)^{-1}\boldsymbol{H}_s$ 中.并且将 $\exists i,(\boldsymbol{H}_s^{i\perp}\boldsymbol{h}_s^*>0)\vee(\boldsymbol{H}_s^{i\perp}\boldsymbol{h}_s^*<0)$ 定义的所有不等式添加到 ILP 的约束条件中,以保证 $\boldsymbol{h}_s^*$(寻找的下一行)在 $\boldsymbol{H_s}$ 的正交子空间中至少有一个非零部分.求解 ILP 问题,得到构成循环子块的下一个超平面.如此迭代,直到求出构成循环子块的所有超平面.

完成一次程序仿射划分后,针对多级存储层次,重新选择当前关注元,再次进行程序仿射划分.例如:选择除

© 中国科学院软件研究所 http://www.jos.org.cn

关注元 z 以外的权重值最大的关注元 z'.与上述方法类似,针对当前关注元进行相关性分析和下一级的仿射划分,直到多级程序仿射划分完成.

本节基于文献[6]中的算法进行扩展与改进,形成了基于关注元的多级仿射划分算法,描述如下:

算法 1. 基于关注元的多级仿射划分算法.

//输入:核心循环与体系结构图模型 G_a;

(1) 读取 G_a 中的存储层次信息,确定循环仿射划分参数

依上文方法分别求得程序仿射划分级数及各级循环子块的大小:n,sl_size_1,sl_size_2 和 sl_size_3;

(2) 在程序操作数据集合Ω中,根据数据在程序执行性能中的权重值,选取关注元 z

将选取的关注元 z 设为当前关注元;

对当前关注元 z 进行数据相关性分析,得到多面体相关图模型 PDG^z;

$\Omega \leftarrow (\Omega - z)$;

(3) 求解构成循环子块的第 1 个超平面

初始化超平面的约束集合 $C=\phi$,目标函数 $OF=\phi(\boldsymbol{i}_t)-\phi(\boldsymbol{i}_s)$;

//求超平面的约束条件集合

for PDG^z 中的每一条边 e

分析得到:该数据相关性对应的超平面合法的必要条件 $C_{e1}(i)$;

分析得到:该超平面处为目标函数 OF_1 最小上界的条件 $C_{e2}(i)$;

计算 $C=C\cup C_{e1}(i)\cup C_{e2}(i)$;

以约束 C 和目标函数 OF_1 为参数,求解 ILP 问题,得到第 1 个超平面;

(4) 迭代求解其余超平面

while (已知超平面数<构成循环子块的超平面数)

将已求解超平面作为一行,加入矩阵 $\boldsymbol{H}_S$;

计算 $\boldsymbol{H}_S$ 的正交子空间为 $\boldsymbol{H}_S^{\perp} = \boldsymbol{I} - \boldsymbol{H}_S^T(\boldsymbol{H}_S\boldsymbol{H}_S^T)^{-1}\boldsymbol{H}_S$;

设置下一个超平面的约束 C':(a) 必须在正交子空间 $\boldsymbol{H}_S^{\perp}$;(b) 保证下一个超平面非零;

计算 $C=C\cup C'$;

求解 ILP 问题,得到下一个超平面;

求得当前级别的循环子块,并分析得到当前循环子块的执行时间;

(5) 求解下一级别的循环子块

if 当前级数<n

转到步骤(2)执行;

else 多级循环子块求解结束;

//输出:各并行任务(即循环子块)与各任务在各处理器核上的执行时间.

3.3 “循环子块——MPSoC各处理器核”的映射

在第 2.3 节中,已将“循环子块-MPSoC 各处理器核”映射问题定义为图节点多着色问题.对于复杂应用而言,该问题的解空间巨大,仅借助设计经验与人工分析难以获得满意的映射方案.因此,使用 MMAS(max-min ant system)算法[13]进行组合寻优.选择该算法的原因为是:(1) 对于多着色问题,蚁群优化算法可以在合理的运行时间内,找到接近最优的高质量解[14];(2) 相对于经典的组合优化算法,在解决 TSP(traveling salesman problem)和 QAP(quadratic assignment problem)问题时,MMAS 被证明具有最好的性能[13].而图节点的多着色问题是 TSP 在映射问题上的变种形式.

解决多着色问题时,MMAS 的相关规则定义如下:

- 着色对象:PIA-CFTG 模型 G_f 的节点.设循环子块的数目为 n,处理器核的数目为 m,则 G_f 节点数目为

© 中国科学院软件研究所 http://www.jos.org.cn

(n+2),每个节点着色为 m 种颜色中的一种;

- 目标函数、约束条件与适应度函数:着色过程中,目标函数 OF_2 与约束条件见定义 5.适应度函数定义为 $Fitness=1/OF_2$;
- 图节点的着色策略:对于除结束节点以外的每个节点 t_i,蚂蚁 k 试图确定 t_i 的每个后继 t_j 的颜色.确定的依据为边 e_{ij} 的全局启发信息 $\tau_{ij}^k(t)$ 和节点 t_j 的局部启发信息 $\eta_{ij}^k(t)$.其中,$\tau_{ij}^k(t)$ 是 t 时刻边 e_{ij} 上残留的信息素浓度.$\eta_{ij}^k(t)=1/OF_2$ 是 t_j 被着色为 c_k 的局部启发信息.

在时刻 t,节点 t_i 上的蚂蚁 k,猜测后继节点 t_j 被着色为 c_k 的概率为

$$p_{ij}^k(t)=\frac{[\tau_{ij}^k(t)]^\alpha[\eta_{ij}^k(t)]^\beta}{\sum_{s=1}^{m}[\tau_{is}^k(t)]^\alpha[\eta_{is}^k(t)]^\beta},$$

其中,α和β是全局启发信息和局部启发信息的权重因子.

实际着色时,根据所有蚂蚁的着色判断,确定节点 t_j 被着色为 c_k 的概率为

$$p_i^k(t)=\frac{\text{猜测}t_i\text{猜测着色为}c_k\text{的前驱节点个数}}{\text{节点}t_i\text{的前驱节点个数}}.$$

- 信息素更新策略:每次迭代过程中,只有找到当前迭代中最优解的那只蚂蚁更新信息素.信息素 $\tau_{ij}^k(t)$ 的更新函数为

$$\tau_{ij}^k(t+1)=\begin{cases}\tau_{ij}^k(t)_{\max}, & \text{if } \tau_{ij}^k(t)>\tau_{ij}^k(t)_{\max}\\ \tau_{ij}^k(t)_{\min}, & \text{if } \tau_{ij}^k(t)<\tau_{ij}^k(t)_{\min}\\ (1-\rho)\tau_{ij}^k(t)+\Delta\tau_{ij}^k(t)_{best}, & \text{otherwise}\end{cases},$$

其中,ρ是信息素挥发率; $\tau_{ij}^k(t)_{\max}=\sum_{i=1}^{t}(1-\rho)^{t-i}\frac{1}{OF_2(s^{gb})}+(1-\rho)^t\tau_{ij}^k(0)$; $\tau_{ij}^k(t)_{\min}=\frac{\tau_{ij}^k(t)_{\max}(1-p_{dec})}{\left(\frac{n}{2}-1\right)p_{dec}}$; s^{gb} 为全局最优解;p_{dec} 表示所有着色点选择对应颜色的概率,设为常数; $\Delta\tau_{ij}^k(t)_{best}$ 是本次迭代最佳蚂蚁对边 e_{ij} 上 c_k 信息素的增加量,根据 Ant-Cycle 模型,定义如下:

$$\Delta\tau_{ij\,best}^k=\begin{cases}\frac{Q}{OF_2(s^{best})}, & \text{if 蚂蚁}k\text{在本次循环中将}e_{ij}\text{着色为}c_k\\ 0, & \text{otherwise}\end{cases},$$

其中,Q 为常数,$OF_2(s^{best})$表示 s^{gb}(全局最佳)或 s^{ib}(本次迭代最佳)选择着色方案的目标函数值.

根据上述规则,求解 MPSoC 应用映射问题的 MMAS 算法描述如下:

算法 2. PIA-CFTG 模型 G_f 的节点多着色算法.

//输入:PIA-CFTG 模型 G_f(节点数 n+2);

1) 初始化 MMAS 参数

设置参数 Q,α,β,ρ;

设蚂蚁数 $m=n+2$,并将 m 只蚂蚁随机置于 G_f 节点上;

禁忌表的初始值 $tabu_k=\phi$;禁忌表索引 $s=1$;

设置时间 $t=0$,循环迭代次数 $N_c=0$,最大循环次数为 N_{cmax};

设置每条边(i,j)的初始信息量 $\tau_{ij}^k(0)$ 为常数,初始信息量增量 $\Delta\tau_{ij}^k(0)=0$;

2) 着色方案的寻优过程

while (禁忌表未满)

设置 $s=s+1$;

for G_f 的每个节点

© 中国科学院软件研究所 http://www.jos.org.cn

for 该节点上的每只蚂蚁,设为蚂蚁 k

(1) 在约束条件下,以概率 $p_i^k(t)$ 选择颜色;

(2) 将蚂蚁 k 移动到下一个 G_f 节点;

(3) 更新局部信息素;

将刚刚经过的 G_f 节点加入禁忌表中;

计算 $\Delta\tau_{ij\,best}^k$;

更新全局信息素;

if 当前的循环迭代次数 $N_c<N_{cmax}$

清空所有的禁忌表;

设置 s=1,t=t+1;

for G_f 的每个节点

for 该节点上的每只蚂蚁,设为蚂蚁 k

$tabu_k(s)$=1(一次循环后蚂蚁又重新回到初始位置)

返回步骤 2);

else 输出最优适应函数值;

//输出: G_f 节点的高质量着色方案;

4 应用实例与结果分析

4.1 应用实例与实验环境

在应用实例中,异构 MPSoC 采用课题组开发的 ESP-MPSoC.ESP-MPSoC 在第 2.2 节中已有介绍,这里不再赘述.所采用的典型应用算法来自于多媒体处理领域,包括矩阵向量转置(matrix vector transpose,简称 MVT)、LU 分解、离散余弦变换(discrete cosine transform,简称 DCT)、综合应用算法(MVT,LU 和 DCT).其中,LU 分解是图像识别等图像处理算法中的频繁操作,DCT 是图像压缩和视频处理算法的核心计算,MVT 更是图像与视频处理中的基本运算.在多媒体处理 MPSoC 的应用中,这些算法频繁出现且计算复杂性高.优化这些算法的处理性能,对于提高 MPSoC 多媒体处理性能与效率具有重要作用.

实验环境及参数设置如下:

- 硬件配置:AMD Athlon 4400+(2.31GHz),2G 内存;
- 操作系统:Windows XP Professional SP3;
- 应用软件:(1) ARM Profiler 2.0.该软件用于测试应用在 ARM1136JF-S 上的运行时间.模拟运行过程中,时钟频率设置为 200MHz,启用向量浮点部件;(2) ModelSim SE 6.1f.该软件用于分析优化前后的应用算法在 SP 与异构 MPSoC 上的运行时间.模拟运行过程中,时钟频率设置为 200MHz;(3) 编译器.编译 ARM1136JF-S 程序时,使用 RealView Development Suite v4.0 中提供的 armcc 编译器,采用默认的优化选项.其中,优化级别为 O2.编译 SP 程序时,使用课题组开发的编译器原型.

4.2 实验结果与分析

针对各种应用算法,使用第 3.2 节的方法进行多级仿射划分,产生多级循环子块.表 1 给出经过多级仿射划分后,单个最内层循环子块在各处理器核上的执行时间.这些数据是使用 ModelSim 模拟分析得到的,这些执行时间也是“循环子块-MPSoC 各处理器核”映射的主要依据.其中,在 DCT 应用优化过程中,为了更好地加速 DCT 处理,将 DCT 分为 3 个部分(DCT-part1,DCT-part2 和 DCT-part3)分别加以处理.另外,EstarIII 为异构 MPSoC 中的 32 位 RISC 通用微处理器核,*SP*1 和 *SP*2 分别是异构 MPSoC 中的两个专用指令集处理器核,如图 1 所示.

© 中国科学院软件研究所 http://www.jos.org.cn

Table 1 Execution time of single inner-most loop sub-blocks on each processor

表 1 单个最内层循环子块在各处理器上的执行时间

Algorithms	Execution time (ns)		
	EstarIII	*SP*1	*SP*2
DCT-part1 (*N*=2000)	720.000	1 366.787	1 776.822
DCT-part2 (*N*=2000)	5 156.000	1 125.390	1 395.483
DCT-part3 (*N*=2000)	22 375.000	3 044.942	4 323.818
LU (*N*=2000)	27 400.000	5 535.170	3 690.113
MVT (*N*=10000)	17 626.000	4 073.127	2 789.813

从表 1 的数据可以看出,*SP*1 和 *SP*2 的处理时间普遍上要比 EstarIII 的处理时间要短.主要是因为 *SP*1 和 *SP*2 具有大位宽(128 位)的访存与计算能力以及 SIMD 并行计算模式,并分别具有针对图像和视频处理的专用指令.因此,*SP*1 和 *SP*2 上的处理时间比 EstarIII 的处理时间要短.

而 DCT-part1 是一个例外.其在 EstarIII 上的处理时间较短.原因是该实例仅包含简单的赋值操作,而没有其他计算,无法发挥 *SP*1 和 *SP*2 的计算能力.相反地,如果使用 *SP*1 或 *SP*2 处理该实例,反而需要在 EstarIII 和 SP 间传输大量的数据,消耗了额外的时间.因此,DCT-part1 在 EstarIII 上的处理时间较短.

从表 1 所示数据可以看出,同一个循环子块,在 *SP*1 和 *SP*2 上的执行时间有所不同.主要是因为:虽然 *SP*1 和 *SP*2 都具有大位宽(128 位)的访存与计算能力以及 SIMD 并行计算模式,但是它们在指令集上仍然存在部分差异.这种差异导致了 *SP*1 和 *SP*2 的处理时间有所不同.

多级仿射划分后的应用算法,分别在单 SP 和异构 MPSoC 上运行,并测得执行时间.其中,应用算法在异构 MPSoC 上运行之前,需要使用第 3.3 节中提到的方法将各个循环子块映射到 MPSoC 的各处理器核上.表 2 给出了将应用算法映射到单 SP 和异构 MPSoC 上的执行时间.其中,这里的单 SP 采用 *SP*1;综合应用的单处理器核处理时间为其所包含应用的处理时间的累加.通过比较优化前后的应用算法在单个 SP 上的运行时间,验证多级仿射划分对于改善性能的效果;通过应用算法在异构 MPSoC 上的运行时间,验证多级仿射划分与“应用-MPSoC 各处理器核”映射方法对于改善性能的效果.

因为异构 MPSoC 由课题组自主开发,单 SP 和异构 MPSoC 上应用算法的执行时间不能给出很直观的印象.这里将各应用算法在 ARM1136JF-S 上的执行时间作为参考点,使应用算法在单 SP 和异构 MPSoC 上的处理性能优化效果更加直观.ARM1136JF-S 采用 ARMv6 指令集,拥有 8 级流水线和高带宽存储系统,集成了浮点运算和 DMA 功能,适用于处理嵌入式多媒体信息.

Table 2 Execution time of each algorithm on every processor before and after optimization (ns)

表 2 优化前后的各应用算法在不同处理器上的执行时间 (ns)

Algorithms	ARM1136JF-S	Single SP before optimization	Single SP after optimization	MPSoC after optimization
DCT (*N*=2000)	801 601 192 043	1 232 646 450 000	335 526 807 060	191 556 841 823
LU (*N*=2000)	338 870 804 830	498 514 673 843	174 051 210 938	108 782 006 837
MVT (*N*=10000)	11 815 777 005	2 364 000 000	2 140 588 542	1 337 867 840
Mixed algorithms (DCT, LU, MVT)	1 152 287 773 878	1 733 525 123 843	511 718 606 540	302 385 620 745

表 2 中的第 2 列表示各应用算法在 ARM1136JF-S 上的执行时间.因为 ARM 应用广泛,将 ARM 处理时间作为参考点,可以使 SP 和异构 MPSoC 的处理时间更直观地被理解.ARM 的处理时间是使用 ARM Profiler 2.0 测试得到的.

表 2 中的第 3 列表示未经多级仿射划分的应用算法在单个 SP 上的执行时间.SP 具有大位宽(128 位)的访存与计算能力以及 SIMD 并行计算模式,并具有针对多媒体处理的专用指令集.因此,单 SP 在性能上接近于 ARM1136JF-S.然而,没有经过多级仿射划分的应用算法,在数据局部性与数据传输效率方面没有进行优化.虽然可以进行 SIMD 并行转换,但应用优化前的数据相关性阻碍了部分 SIMD 并行执行的机会.因此,应用优化前的单 SP 执行性能接近于 ARM1136JF-S,但仍有较大的性能改善空间.

表 2 中的第 4 列表示经过多级仿射划分后的应用算法在单个 SP 上的执行时间.多级仿射划分引入了存储

© 中国科学院软件研究所 http://www.jos.org.cn

层次特征,进行与存储层次相对应的多级仿射划分.相对于优化前的应用算法,在数据局部性与数据传输效率方面进行了优化.而且,仿射划分优化了应用算法中固有的数据相关性,更有利于发挥单 SP 的 SIMD 并行处理能力.由实验结果可以看出,应用算法在多级仿射划分后使单 SP 的处理时间更短.

表 2 中的第 5 列表示经过多级仿射划分与“循环子块-MPSoC 各处理器核”映射的应用算法在异构 MPSoC 上的执行时间.在多级仿射划分后,使用组合寻优算法将各个循环子块合理地分配到 MPSoC 各处理器核上,实现负载平衡与优化的处理性能.经过多级仿射划分与映射之后,异构 MPSoC 可以高效地使用 3 种并行加速方法:SIMD 并行、多处理器核并行以及数据并行传输.异构 MPSoC 硬件处理能力与本文优化方法使得异构 MPSoC 的处理时间最短.

图 2 是表 2 的另外一种表示形式.该图给出了优化前后各应用算法在不同处理器核上的加速比(以 ARM1136JF-S 为参考点).

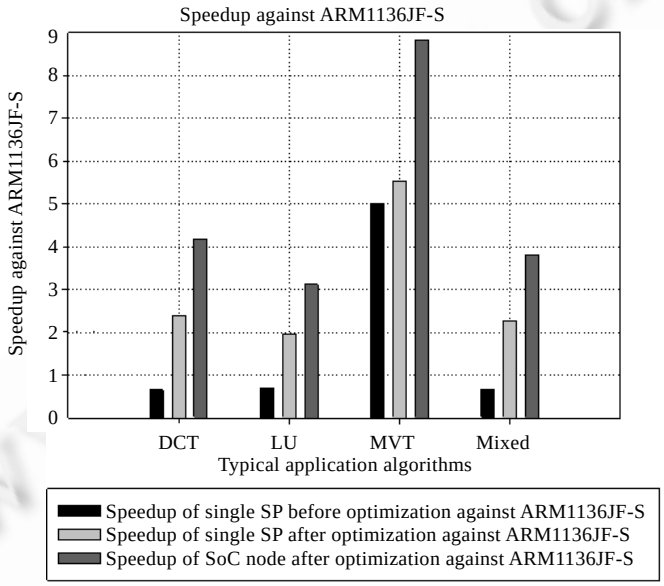

Fig.2 Speedup of algorithms before and after optimization against ARM1136JF-S

图 2 优化前后各应用算法在不同处理器核上的加速比(以 ARM1136JF-S 为参考点)

分别使用 PluTo 与本文方法对应用算法进行优化处理.表 3 分别给出两种方法优化后的应用算法在异构 MPSoC 上的执行时间.

Table 3 Execution time of algorithms running on heterogeneous MPSoC using the proposed approach and PluTo approach (ns)

表 3 使用本文方法与 PluTo 方法优化后的应用算法在异构 MPSoC 上的执行时间 (ns)

Algorithms	PluTo approach	The proposed approach
DCT-part1 (*N*=2000)	27 179 025	26 464 298
DCT-part2 (*N*=2000)	106 331 250	34 265 025
DCT-part3 (*N*=2000)	839 653 125 000	191 496 112 500
DCT (*N*=2000)	839 786 635 275	191 556 841 823
LU (*N*=2000)	630 505 371 095	108 782 006 837
MVT (*N*=10000)	5 859 759 116	1 337 867 840
Mixed (DCT, LU, MVT)	1 476 143 149 617	302 385 620 745

表 3 表明,相对于 PluTo 方法,本文方法获得了明显的性能改进.主要原因为:(1) PluTo 没有考虑异构 MPSoC 的存储层次信息,而本文方法进行了与存储层次相对应的多级仿射划分,优化了各存储层次的数据传输与存储

© 中国科学院软件研究所 http://www.jos.org.cn

性能;(2) PluTo 针对某一个固定数组进行仿射划分,而本文方法采用基于关注元的仿射划分,分析数组在应用性能中的权重值,并动态地选择关注元,可以进行针对多个数据的相关性、数据局部性的优化;(3) PluTo 提供了一个应用优化的通用框架,然而对于异构的 MPSoC,并没有提供有效的“应用-MPSoC 各处理器核”映射策略,没有考虑存储层次特征和各处理器核的差异性,导致各处理器核的负载失衡.针对上述问题,本文方法引入了异构 MPSoC 的体系结构模型,采用基于关注元的多级仿射划分,并使用组合寻优算法进行映射空间探索,最终获得了明显的性能提高.

其中,对于 DCT-part1,本文方法与 PluTo 方法获得了相近的执行时间.主要原因为:该应用在各个处理器核上的执行时间差异较小,再加上问题规模限制了映射探索的空间大小,使得通过组合寻优算法得到的映射方案与 PluTo 选择的映射方案相近,最终仅使本文方法相对于 PluTo 方法取得了微弱的优势.

图 3 是表 3 的另外一种表示形式,该图给出了本文方法相对于 PluTo 方法的加速比.

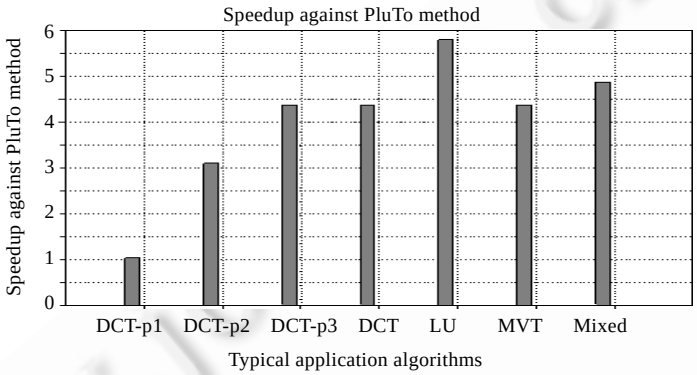

Fig.3 Speedup against PluTo of the proposed approach

图 3 本文方法相对于 PluTo 方法的加速比

上述实验以异构 MPSoC 和典型多媒体处理算法为实例,验证了本文方法的有效性.实验结果表明,该方法明显地改善了典型多媒体处理算法在异构 MPSoC 上的处理性能.正如前文所言,这些典型算法是多媒体应用中频繁操作的耗时部分.对这些算法的性能进行优化,对于提高 MPSoC 的多媒体处理能力与效率具有重要作用.

5 总结与展望

本文提出了一种异构 MPSoC 的应用算法性能优化方法.该方法通过应用特征分析、循环仿射划分、“应用功能-MPSoC 各处理器核”映射 3 个步骤,获得优化的数据局部性与多级并行,从而明显改善了异构 MPSoC 上多媒体处理算法的执行性能.

本文目前集中于研究异构 MPSoC 的性能优化问题,然而对于嵌入式多媒体处理 MPSoC 而言,功耗是另外一个非常重要的设计目标与评价标准.今后将进一步针对 MPSoC 功耗优化问题进行专门研究.

References:

[1] Artieri A, Alto VD, Chesson R, Hopkins M, Rossi MC. Nomadik: Open multimedia platform for next-generation mobile devices. 2003. http://www.st.com

[2] Dutta S, Jensen R, Rieckmann A. Viper: A multiprocessor SOC for advanced set-top box and digital TV systems. IEEE Design & Test of Computers, 2001,18(5):21−31. [doi: 10.1109/54.953269]

[3] Albani L, Chiesa P, Covi D, Pedegani G, Sartori A, Vatteroni M. VISoC: A smart camera SoC. In: Proc. of the 28th European Solid-State Circuits Conf. 2002. 367−370. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1471541

© 中国科学院软件研究所 http://www.jos.org.cn

[4] Yan M, Li SK, Shen JL, Xie L, Liu L, Wan GW, Yin LZ. A heterogeneous multicore SoC optimized for embedded visual media process. In: Proc. of the WRI Int'l Conf. on Communications and Mobile Computing. Kunming, 2009. 12–16. http://www.computer.org/portal/web/csdl/doi/10.1109/CMC.2009.112 [doi: 10.1109/CMC.2009.112]

[5] Lu SL, Zhao Z. Research on nodes technology in wireless sensor networks. Information Technology Letter, 2008,6(3):13−19 (in Chinese with English abstract).

[6] Bondhugula UKR. Effective automatic parallelization and locality optimization using the polyhedral model [Ph.D. Thesis]. Chicago: The Ohio State University, 2008.

[7] Pouchet LN, Bastoul C, Cohen A. LetSee: The LEgal transformation spacE explorator. In: Proc. of the 3rd Int'l Summer School on Advanced Computer Architecture and Compilation for Embedded Systems. 2007. 247–251. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.97.2036

[8] Lim AW, Lam MS. Maximizing parallelism and minimizing synchronization with affine partitions. Parallel Computing, 1998, 24(3-4):445−476. [doi: 10.1016/S0167-8191(98)00021-0]

[9] Xia J, Yang XJ. A data space fusion based approach for global computation and data decompositions. Journal of Software, 2004, 15(9):1311−1327 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/1311.htm

[10] Schrijver A. Theory of Linear and Integer Programming. New York: Wiley Press, 1986.

[11] Zhao P, Li SK, Wang DW, Yan M. Application-Driven system-on-chip system model extraction approach. In: Proc. of the 12th Int'l Conf. on Computer Supported Cooperative Work in Design. 2008. 123–128. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.97.2036 [doi: 10.1109/CSCWD.2008.4536968]

[12] Vivekanandarajah K, Pilakkat SK. Task mapping in heterogeneous MPSoCs for system level design. In: Proc. of the 13th IEEE Int'l Conf. on Engineering of Complex Computer Systems. 2008. 56–65. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4536968 [doi: 10.1109/ICECCS.2008.18]

[13] Stützle T, Hoos HH. Max-Min ant system. Future Generation Computer Systems, 2000,16(9):889−914.

[14] Wang G, Gong WR, Derenzi B, Kastner R. Exploring time/resource trade-offs by solving dual scheduling problems with the ant colony optimization. ACM Trans. on Design Automation of Electronic Systems, 2007,12(4):46–52. [doi: 10.1145/1278349.1278359]

附中文参考文献:

[5] 陆世龙,赵泽.无线传感器网络的节点技术.信息技术快报,2008,6(3):13−19.

[9] 夏军,杨学军.基于数据空间融合的全局计算与数据划分方法.软件学报,2004,15(9):1311−1327. http://www.jos.org.cn/1000-9825/15/1311.htm

赵鹏(1979－),男,河南鹤壁人,博士生,主要研究领域为嵌入式系统与 SoC 设计方法.

李思昆(1941－),男,教授,博士生导师,CCF 高级会员,主要研究领域为嵌入式系统与 SoC 设计方法,虚拟现实与可视化.

严明(1981－),男,博士生,CCF 学生会员,主要研究领域为 SoC 系统结构与编译.

© 中国科学院软件研究所 http://www.jos.org.cn