ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, Supplement, December 2010, pp.290−297

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种面向软件管理内存层次的简易数据分块方法*

刘 勇+, 陆林生, 何王全

(江南计算技术研究所,江苏 无锡 214083)

Easy Data Tiling Method for Software-Managed Memory Hierarchies

LIU Yong+, LU Lin-Sheng, HE Wang-Quan

(Jiangnan Institute of Computing Technology, Wuxi 214083, China)

+ Corresponding author: E-mail: liuli8149@163.com

Liu Y, Lu LS, He WQ. Easy data tiling method for software-managed memory hierarchies. *Journal of Software*, 2010,21(Suppl.):290−297. http://www.jos.org.cn/1000-9825/10030.htm

Abstract: Many-Core CPU preferring software-managed memory hierarchies than the Cache memory hierarchies owe to the area and power consumption. Software-Managed memory hierarchies need soft explicitly managed the data placement and data transfer. This paper proposes an easy data tiling method compilation techniques for these large memory objects such as large arrays base on the program loop characteristic and the scale of program data. This method is easy carry out in the compiler and has the equal efficient as the loop and data tiling method. The experimental results of several benchmarks show that this method can outperform the loop and data tiling method when this method may acquire additional data locality in the memory on chip.
Key words: software-managed memory hierarchy; data tiling; locality optimization

摘 要: 考虑到硬件管理 Cache 多级存储结构在功耗和面积方面的开销过大,众核处理器倾向于采用软件管理的多级存储结构,这就需要软件规划好程序的数据在各级存储上的布局和传输.尝试了一种依赖程序原有循环结构和问题规模的简易数据自动分块方法,根据循环层内的数据访存范围进行相应的分块,避免数据复杂的依赖关系分析,使得该方法易于在编译器中实现.同时可根据需要进一步结合程序变换如循环交换、循环联合和循环分裂等方法得到更佳的分块参数.实验结果表明,在大多数问题规模下与一般分块方法的优化性能相当,但在某些特定问题规模下能够获得较高的优化性能.
关键词: 软件管理存储层次;数据分块;局部性优化

长期以来,程序员在传统通用处理器上享受着硬件管理 Cache 的多层次存储系统带来的编程便捷性和程序运行的高效性.然而随着 CMP 技术的兴起,处理器片上核数的增加,硬件管理的多层次存储系统出现扩展性差和高功耗问题,使得其不适合在众核处理器上采用.一些计算机专家认为,在拥有上千核的众核处理器上设计一个硬件管理的多层次存储系统基本上是不可能的[1,2].Intel 的 Larrabee 处理器项目被取消了,部分原因是难以设计一个有效的基于硬件管理 cache 的多层次存储系统来满足处理器上众多计算核的访存需求.由于硬件管理多层次存储系统在众核处理器上的实现过于困难,硬件工程师们尝试了其他的实现途径——软件管理的多层

* Supported by the National Basic Research Program of China under Grant No.2007CB310900 (国家重点基础研究发展计划(973))
Received 2010-06-15; Accepted 2010-12-10

次存储系统(software-managed memory hierarchies system,简称 SMMHS).在该存储系统中,每个计算核或者某几个计算核拥有一块分别编址的片上存储器,其完全由用户或者说软件来管理的,该存储器拥有比片外内存更快的访存速度和更宽的访存带宽,如,cyclops64 处理器[3]每个计算核上的 scratch-pad、Cell BE 处理器[4]SPE 上的 local store,Nvidia GT200[5]上每个 SM 上的 Shared Memory 和 FT 处理器[6]上的 stream register file(SRF)等.片上存储器与片外存储器之间一般还拥有硬件支持的异步数据块传输通道如 DMA.

SMMHS 需要程序员编程时仔细考虑程序的数据在各级存储器的放置以及数据在各级存储器之间的移动来获得最佳程序性能,充分挖掘众核处理器的硬件潜力.程序员在开发并行应用被诸如数据分布与依赖、计算划分、同步和通信等困难问题上已经耗费了大量精力,如果还要考虑数据在各级存储器中存储和移动等问题,那么程序员的编程负担将变得极为繁重,导致其编程生产率低下、应用软件的开发成本急剧增加.更加地,由于并行调试和性能调优工具的欠缺,在拥有 SMMHS 众核处理器上开发出一个正确而充分发挥硬件潜力的并行应用,对普通程序员而言,变得越来越遥不可及.因此,亟需一些能够自动管理程序数据在 SMMHS 的存储和传输的优化方法和软件工具.

1 相关的工作

自动管理程序数据在 SMMHS 中的存储和传输的优化方法近几年成为高性能众核处理器上编译优化技术的研究热点,出现了一大批的研究文献[7−20].主要有如下几个分类:

(1) 编译指导的数据布局技术

文献[7]利用图着色的方法在 FT 流处理器上对科学计算应用的循环核心的数据布局和传输进行自动优化.文献[8]利用区间着色的方法对 FT 流处理器上的 SRF 进行自动分配和数据间的传输优化,文献中改进了区间着色的前提条件只要数组的生存周期相干图满足可比图性质即可实现数据在 SRF 最佳分配.文献[9]阐述了利用图着色法在 FT 流处理器上的程序数据在 SRF 分配优化实现.

(2) 软件模拟 Cache 技术

文献[10−13]主要研究 CellBE 处理器上的 local store 的自动优化问题,采用基本是软件模拟 cache 的思想,实现了一个高效的软件模拟 cache 优化程序中的数据访问[10],优化了利用静态缓冲的 DMA 数据传输[11],高效预取非规则访问的数据[12]以及利用编译信息对 DMA 进行合理的编排优化[13]等.

(3) 自动分块技术

分块技术是提高传统处理器的 Cache 上访存效率的有效方法[20−23],也是在 SMMHS 上挖掘系统性能的有效手段[24,25].但是二者之间有一些差别,前者只需要考虑如何循环分块使得已驻留在 Cache 中的数据被尽可能多的被访问.而后者不但需要考虑如何循环分块,还需考虑数据的分块问题即片上内存容量大小对数据分块大小的限制,而且循环分块和数据分块之间还要相互匹配.文献[24]提出了在管理多层次存储系统上程序中数据分块结合循环分块的一些基本方法.文献[25]提出了一种基于参数模型的长流分段技术,其首先建立了一个预取和重用优化指导的参数模型,反映段大小对流处理器上程序性能的影响;基于该模型分别研究了计算密集型程序和访存密集型程序的最优分段策略,提出一种面向任意程序的最优分段技术.

(4) 综合性的方法

既进行程序变换如循环分块,又结合数据布局方法的一些优化技术.文献[26]提出了循环分块、数组分块技术与区间着色数据布局技术的结合的优化方法.文献[14]利用多面体模型对需要显式管理多级存储资源的系统上应用实现了自动化管理和相关的优化方法.首先借助多面体模型的数据访问空间给每个数组变量创建片上内存缓冲区放置循环中计算所需的部分数据,自动分析数据的访问模式,将数据访问下标空间进行转换,同时产生各个存储层次之间的数据传输语句.文献[15]提出了一个优化 GPU 上通用科学计算仿射嵌套循环访存的编译器框架,其同样采用了多面体的理论,对仿射嵌套循环的.文献[16−18]主要研究了 cyclops64 处理器上的一种基于自动分块的 OpenMP 编译优化技术.

数据自动分块技术最基本的目的就是解决大于片上内存容量的数组如何利用片上内存进行最佳的访存优

化,一般的思路是进行循环分块然后数组变量根据循环分块进行.为了获得程序的最佳性能,软件需要考虑程序中的数据访问对离计算核近,速度快的片上存储器的有效利用.对程序中大于片上存储容量的数据内存对象例如大数组变量,一般需要分析数据的依赖关系,结合程序变换方法将循环进行分块,确保每个循环块中的数据能够在片上内存布局下,这种分块方法实现难度大而且不能充分挖掘原有循环结构的局部性.本文尝试另辟途径,先从原有循环结构层次中的语句访问数据规模出发,在片上内存容量满足时,则直接利用原有循环结构对数组进行分块,无需进行循环分块程序变换也即不改变原有的循环迭代顺序,然后直接在循环中插入数据传输语句.因而就不需要分析迭代间语句之间的数据依赖关系,但是迭代内的数据依赖关系.只有当容量不满足时,才采用循环分块技术来进行相应的循环变换.同时当原有循环结构经分析不具有良好的局部性或者竞争资源的内存对象过多,可以尝试对其进行相关的程序变换例如:循环交换,循环分裂等.这种简易的方法有时能够带来额外的数据局部性而比直接结合循环分块方法的性能要更佳些,几个基准程序循环的实验证实这一点.

2 简易数据自动分块方法

2.1 基本概念和模型

数据自动分块是在循环中进行的,首先对循环的基本概念介绍如下:

1) 嵌套循环:一个 n 层嵌套循环 L 的迭代空间是一个 n 维有限凸多面体 Z^n,其由各层循环的边界约束条件构成.循环中的每次迭代对应与多面体的一个整数格点,它由迭代向量 $\vec{i}=(i_1,i_2,...,i_n)$ 唯一确定,其中 i_k 是嵌套循环中的从外往里数的第 k 个循环迭代变量.

2) 嵌套循环信息表示:程序中一个嵌套循环 L 所有信息可以用 $L=\langle S,\delta,\eta,D,F,\omega\rangle$ 来表示,S 是循环 L 中程序语句的有序列表,这些语句按照其在程序文中的位置排序.δ_s 是语句 s 的循环深度,即包围语句 s 的循环个数.$\eta_{s,s'}$ 是语句 s,s' 的公共循环深度,即包围语句 s,s' 的公共循环个数. $D_s(\vec{i})$ 表示语句 s 是否是有效循环迭代的线性约束,长度为 δ_s 的迭代向量 $\vec{i}$ 是语句 s 一个有效迭代实例,当且仅当它满足 $D_s(\vec{i})\geqslant\vec{0}$, 约束条件可从包围语句 s 的循环上下界推导出,每个约束条件是循环迭代变量和循环不变参数的仿射表达式. $F_{szr}(\vec{i})$ 是语句 s 中数组 z 的第 r 个数组引用的仿射函数,仿射函数将一个迭代向量 $\vec{i}$ 映射到一个 m 维的数组访问向量,其一般是循环索引变量和循环不变参数的仿射表达式.ω_{szr} 是一个描述语句 s 中数组 z 的第 r 个数组引用的是否是写操作的布尔值,当该引用是写操作是,ω_{szr} 值为真.

3) 循环层的数据访存范围:设嵌套循环 L 中每个语句 s 在包围其 δ_s 层循环中的每层循环的数据访存范围 $DR_i(s)(1\leqslant i\leqslant\delta_s)$,那么循环 L 中 n 层循环第 i 层的数据空间需求为 $\bigcup_{s\in S}DR_i(s)$, 这里,对每个语句 s 有 $n\geqslant\delta_s$,因此,嵌套循环中,当第 i 层循环的 $i\geqslant\delta_s$ 时,令 $DR_i(s)=\phi$ or $DR_{\delta_s}(s)$.

当空间较为充裕时,则为语句 S 保留在其最末层循环访问的数据范围空间,使得语句 S 的访存效率得以保证,当空间不充裕时,则不为语句 S 保留在其最末层循环访问的数据范围空间,放弃对语句 S 的访存性能优化.

由于片上容量限制,我们的数据分块优化方法遵循如下准则:

准则 1. 两个语句 $s1,s2$ 访问了同一个数组变量 A,当它们在 $\eta_{s1,s2}$ 个公共循环层中访问的数据范围是一致的,那么数组 A 在语句 $s1,s2$ 中被访问的数据区存在重用.如果片上内存能够容纳数组 A 的数据区,那么该数据的重用得以实现,从而则该数据在片上内存中具有时间局部性.

准则 2. 语句 S 中访问的某个数组变量 A 的数据范围在某几个循环层是一致的,那么数组 A 被访问的数据区在这几个循环的迭代间存在重用.如果片上内存能够容纳数组 A 的数据区,那么该数据的重用得以实现,从而则该数据在片上内存中具有时间局部性.

准则 3. 两个语句 $s1,s2$ 访问了同一个数组变量 A,当它们在 $\eta_{s1,s2}$ 个公共循环层中访问的数据范围是一致的,那么数组 A 在语句 $s1,s2$ 中被访问的数据区存在部分重叠,那么如果片上内存能够容纳数组 A 两个数据区的并集,那么该数据的重用得以实现,从而重叠部分的数据在片上内存中具有时间局部性.

2.2 实例研究

根据片上容量限制和应用问题规模来寻求满足第 2.1 节中 3 个准则中局部性条件的数据自动分块技术就是本文方法实现的出发点.为了阐述方法具体实现思路,下面以 seidel 迭代为例进行分析.

按照前面介绍的模型,对于如图 1 所示的 seidel 程序,嵌套循环 L 只包含一个语句 S,S 在各层循环的访存范围见表 1.

```
for (t=0; t<=T−1; t++) {
    for (i=1; i<=N−2; i++){
        for (j=1; j<=N−2; j++){
            a[i][j]=
(a[i−1][j−1]+a[i−1][j]+a[i−1][j+1]+a[i][j−1]+a[i][j]+a[i][j+1]+a[i+1][j−1]+a[i+1][j]+a[i+1][j+1])/9.0;   //S
        }
    }
}
```

图 1　seidel 程序循环

表 1　seidel 程序语句 S 在各层循环的访存范围

循环层	数组 a 的访存范围
t	a[*][*]
i	a[*][*]
j	$a[i][*]\ U\ a[i-1][*] U\ a[i+1][*]$

1) 如果整个数组 a 可以存储在片上内存中,那么循环 t 对数组 a 的重用得以实现,且无须对程序代码进行变换.

2) 否则,根据循环 i 每次迭代需要 $a[i][*]\ U\ a[i-1][*] U\ a[i+1][*]$这 3 个一维长度的子数组.则不难发现循环 i 的迭代间存在数据的重用,循环 i 第 l 次迭代使用的数据$\{a[l][*]\ U\ a[l-1][*]\ U\ a[l+1][*]\}$中 $a[l][*]\ U\ a[l+1][*]$在第 l+1 次迭代中继续使用.因此,如果片上内存能够存储 3 个 a 的一维子数组,则这种重用得以实现,程序变换如图 2 所示.如果这种访存范围的数据区不能再片上内存存储下,那么则需要对嵌套循环进行分块,然后再进行相关的数据分块,二者之间还要匹配相关参数.

```
Local Type aa[3][N];
for (t=0; t<=T−1; t++)
{
    DMA_get(&a[0][0],&aa[0][0],N*sizeof(Type));
    DMA_get(&a[1][0],&aa[1][0],N*sizeof(Type));
    for (i=1; i<=N−2; i++)
    {
        DMA_get(&a[i+1][0],&aa[2][0],N*sizeof(Type));
        for (j=1; j<=N−2; j++)
        {
            aa[1][j] = (aa[0][j−1] + aa[0][j] + aa[0][j+1]
                      + aa[1][j−1] + aa[1][j] + aa[1][j+1]
                      + aa[2][j−1] + aa[2][j] + aa[2][j+1])/9.0;
        }
        memcpy(&aa[0][0],&aa[1][0],N);
        memcpy(&aa[1][0],&aa[2][0],N);
        DMA_put(&a[i][0],&aa[1][0],N*sizeof(Type))
    }
}
```

图 2　seidel 程序循环的数据分块示例 1

3) 当片上容量不能存储下 *a* 的 3 个一维子数组,那么迭代 *i* 间的数据重用性在片上内存中不具有局部性,那么需要对循环 *j* 进行分段,根据准则 3,数组 *a* 某一维上的 3 个访问之间存在访存范围的重叠,则将它们的数据区扩大为 *block*+2,就可以实现某一维上 3 个访问之间在片上内存中的数据局部性.程序变换如图 3 所示.

```
Local Type aa[3][block+2];
for (t=0; t<=T−1; t++) {
    for (i=1; i<=N−2; i++) {
        aa[0][0]=a[i-1][0]; aa[0][0]=a[i−1][1]; aa[1][0]=a[i][0];
        aa[1][0]=a[i][1]; aa[2][0]=a[i+1][0]; aa[2][0]=a[i+1][1];
        for (j=1; j<=N−2; j+=block){
          DMA_get(&a[i−1][j+1],&aa[0][2],block*sizeof(Type));
          DMA_get(&a[i][j+1],&aa[1][2],block*sizeof(Type));
          DMA_get(&a[i+1][j+1],&aa[2][2],block*sizeof(Type));
           for (jj=0;jj<block;jj++){
                aa[1][jj+1]=(aa[0][jj]+aa[0][jj+1]+aa[0][jj+2] + aa[1][jj] + aa[1][jj+1]+aa[1][jj+2]
                        +aa[2][jj]+aa[2][jj+1]+aa[2][jj+2])/9.0;
            }
            DMA_put(&a[i][j],&aa[1][1],block*sizeof(Type));
            aa[0][0]=aa[0][block];      aa[0][0]= aa[0][block+1];
            aa[1][0]=aa[1][block];      aa[1][0]= aa[1][block+1];
            aa[2][0]=aa[2][block];      aa[2][0]= aa[2][block+1];
         }
    }
}
```

图 3 seidel 程序循环的循环分块与数据分块示例 2

2.3 编译器实现基本框架

编译器实现基本框架如图 4 所示.

1. 按照第 2.1 节中的模型分析程序嵌套循环中所有语句在各层循环的数据访存范围.
2. 按照片上容量的限制和问题 *N* 的规模对嵌套循环 *L* 进行不断降级的数据分块可行性判断.
3. 若这个过程可行,则转向 5;若不可行,则转向 4.
4. 进行相关的循环变换例如循环分裂和循环分块,降低循环层中同时竞争片上内存资源的数据对象个数和大小.转向 1.
5. 进行数据分块,插入相关数据传输代码.结束.

图 4 编译器实现框架

3 实验与验证

选取了 matrix-multiply,fdtd-2d,seidel,jacobi-2d 这 4 个科学计算程序循环进行了实验.实验环境为 PlayStation 3 Cell Broadband Engine,3.2GHZ,软件环境为 Yellow Dog Linux release 5.0,Cell CBE SDK2.1.

Cell 上的 Local Store 有 256KB,本文主要研究数据自动分块方法,因此对每个程序固定两个片上内存容量参数,调节课题规模,使得对片上内存容量分别为运行课题内存总需求容量的 1,1/4,1/9,1/16.直接结合循环分块的数组分块方法简称为 Classic Tiling,本文依赖原有循环结构和问题规模的分块方法称为 Our Method.实验数据表明:在大多数问题规模下,两种方法优化的性能相当,但是在某些特定问题规模情况下,Our Method 性能提

升约 10%左右,实验对比效果如图 5 所示.

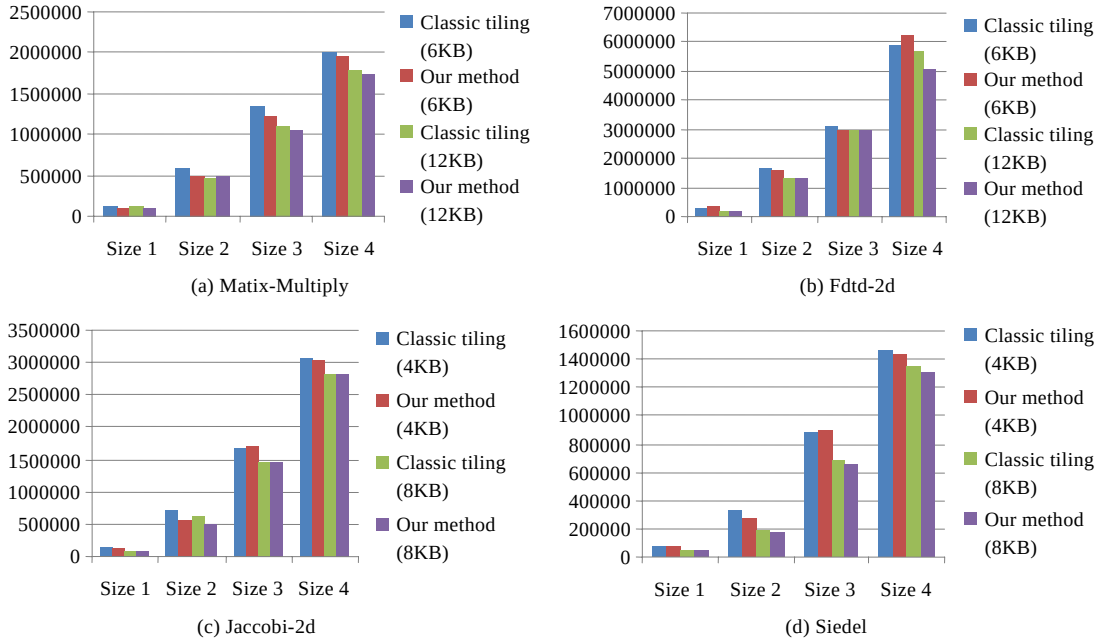

图 5　4 个基准程序循环的自动分块性能对比图

4　结束语

本文提出了一种针对 SMMHS 的简易数据自动分块方法,其依赖程序循环结构和问题规模,以捕捉程序工作数据集在片上内存的局部性优先的准则而进行的数据分块,只有在最低容量需求不被满足时,才结合循环分块等方法进行相应的数据分块.同时本方法易于结合其他程序变换方法做进一步的优化工作,例如当原循环结构经分析不具有良好的局部性或者竞争资源的内存对象过多,可以尝试对其进行循环交换,循环分裂等程序变换.该方法相比较结合循环分块的数据分块方法,在编译器实现上更易于操作,利于程序代码生成.在某些特定的问题规模下,本文方法的性能优化具有更为明显的优势.

进一步的工作考虑将本文简易自动分块方法结合第 1 节提到的基于图着色[7]和区间着色[8]等方法的数据布局技术,在 SMMHS 上探索更有效的程序访存数据流优化技术.

References:

[1] Asanovic K, Bodik R, Catanzaro BC, Gebis JJ, Husbands P, Keutzer K, Patterson DA, Plishker WL, Shalf J, Williams SW, Yelick KA. The landscape of parallel computing research: A view from Berkeley. Technical Report, UCB/EECS-2006-183, Berkeley: EECS Department, University of California, 2006.

[2] Allen F. Compiling for performance a personal tour. 2007. http://awards.acm.org/images/awards/140/vstream/2006/turingaward2006.mov

[3] del Cuvillo J, Zhu W, Hu Z, Gao GR. Towards a software infrastructure for cyclops-64 cellular architecture. In: Proc. of the 20th Int'l Symp. on High-Performance Computing in an Advanced Collaborative Environment (HPCS 2006). 2006.

[4] Kahle JA, Day MN, Hofstee HP, Johns CR, Maeurer TR, Shippy D. Introduction to the cell multiprocessor. IBM Journal of Research and Development, 2005,49(4-5):589−604.

[5] NVIDIA GeForce 8800. http://www.nvidia.com/page/geforce_8800.html

[6] Yang XJ, Yan XB, Xing ZC, Deng Y, Jiang J, Zhang Y. A 64-bit stream processor architecture for scientific applications. In: ISCA 2007: Proc. of the 34th Annual Int'l symp. on Computer architecture. ACM, 2007. 210−219.

[7] Wang L, Yang X, Xue J, Deng Y, Yan X, Tang T, Nguyen QH. Optimizing scientific application loops on stream processors. In: Proc. of the Int'l Conf. on Language, Compilers and Tools for Embedded Systems. 2008. 161−170.

[8] Yang X, Wang L, Xue J, Deng Y, Zhang Y. Comparability graph coloring for optimizing utilization of stream register files in stream processors. In: Proc. of the 14th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming. 2009. 111−120.

[9] Yang XJ, Deng Y, Wang L, Yan XB, Du J, Zhang Y, Wang GB, Tang T. SRF coloring: Stream register file allocation via graph coloring. Journal of Computer Science and Technology, 2009,24(1):152−164.

[10] Gonzalez M, Martorell X, Ayguade E, Sura Z, Chen T, Zhang T, O'Brien K, O'Brien K. A Novel Asynchronous Software Cache Implementation for the Cell-BE Processor. In: Proc. of the 2007 Workshop on Languages and Compilers for Parallel Computing (LCPC 2007). 2007.

[11] Chen T, Sura Z, O'Brien K, O'Brien K. Optimizing the use of static buffers for DMA on a ell chip. In: Proc. of the 19th Int'l Workshop on Languages and Compilers for Parallel Computing. 2006.

[12] Chen T, Zhang T, Sura Z, Tallada MG. Prefetching irregular references for software cache on cell. In: Proc. of the 6th Annual IEEE/ACM Int'l Symp. on Code Generation and Optimization. 2008.

[13] Chen T, Lin H, Zhang T. Orchestrating data transfer for the cell/B.E. processor. In: Proc. of the 22nd Annual Int'l Conf. on Supercomputing, ICS 2008. Island of Kos: ACM, 2008. 289–298.

[14] Baskaran MM, Bondhugula U, Krishnamoorthy S, Ramanujam J, Rountev A, Sadayappan P. Automatic data movement and computation mapping for multi-level parallel architectures with explicitly managed memories. In: Proc. of the ACM SIGPLAN PpoPP. 2008.

[15] Baskaran MM, Bondhugula U, Krishnamoorthy S, Ramanujam J, Rountev A, Sadayappan P. A compiler framework for optimization of affine loop nests for GPGPUs. In: Proc. of the ACM ICS. 2008.

[16] Gan G, Wang X, Manzano J, Gao GR. Tile percolation: An OpenMP tile aware parallelization technique for the cyclops-64 multicore processor. In: Proc. of the 15th Int'l Euro-Par Conf. on Parallel Processing (Euro-Par 2009). 2009.

[17] Gan G, Wang X, Manzano J, Gao GR. Tile reduction: The 1st step towards OpenMP tile aware parallelization. In: Proc. of the Int'l Workshop on OpenMP in a New Era of Parallelism (IWOMP 2009). Berlin, Heidelberg: Springer-Verlag, 2009.

[18] Gan G, Manzano J. TL-DAE: Thread-Level decoupled access/execution for OpenMP on the Cyclops-64 many-core processor. In: Proc. of the 22nd Int'l Workshop on Languages and Compilers for Parallel Computing (LCPC 2009). Springer-Verlag, 2009.

[19] Bondhugula U, Baskaran MM, Krishnamoorthy S, Ramanujam J, Rountev A, Sadayappan P. Affine transformations for communication minimal parallelization and locality optimization of arbitrarily nested loop sequences. Technical Report, OSU-CISRC-5/07-TR43, Ohio State University, 2007.

[20] Bondhugula U, Baskaran MM, Krishnamoorthy S, Ramanujam J, Rountev A, Sadayappan P. Automatic transformations for communication-minimized parallelization and locality optimization in the polyhedral model. In: Proc. of the Int'l Conf. on Compiler Construction (ETAPS CC). 2008.

[21] Song YH, Li Z. New tiling techniques to improve cache temporal locality. ACM SIGPLAN Notices, 1999,34(5):215−228.

[22] Coleman S, McKinley KS. Tile size selection using cache organization and data layout. In: Proc. of the SIGPLAN Conf. on Programming Language Design and Implementation. New York: ACM Press, 1995. 279−290.

[23] Hsu CH, Kremer U. A quantitative analysis of tile size selection algorithms. Journal of Supercomputing, 2004,27(3):279–294.

[24] Zhang C, Kurdahi F. On combining iteration space tiling with data space tiling for scratch-pad memory systems. In: Proc. of the 2005 Conf. on Asia South Pacific Design Automation. New York: ACM Press, 2005. 973–976.

[25] Du J, Ao FJ, Tang T, Yang XJ. Parameter model based strip-mining technique on the stream processor. Journal of Software, 2009, 20(9):2320−2331 (in Chinese with English abstract).

[26] Li L, Wu H, Feng H, JL Xue. Towards data tiling for whole programs in scratchpad memory allocation. In: Proc. of the ACSAC 2007. LNCS 4697, 2007. 63−74.

附中文参考文献:

[25] 杜静,敖富江,唐滔,杨学军.流处理器上基于参数模型的长流分段技术.软件学报,2009,20(9):2320−2331.

刘勇(1981－),男,湖南邵阳人,博士生,研究实习员,主要研究领域为并行语言设计,编译优化.

陆林生(1942－),男,高级工程师,博士生导师,主要研究领域为并行识别与并行计算方法,并行计算环境.

何王全(1975－),男,高级工程师,主要研究领域为并行语言设计,编译优化.