高技术通讯 2010 年 第 20 卷 第 12 期:1229 ~ 1234

doi:10.3772/j.issn.1002-0470.2010.12.004

# 一个面向异构多核处理器 Cell 的资源分配模型①

王 森② 王志英③ 郭贵明

(国防科学技术大学计算机学院 长沙 410073)

**摘 要** 为了充分利用多核处理器提供的多级并行和解决多核资源分配问题,提出了一种将多任务并行程序映射到多核处理器平台上的模型驱动的方法。该方法首先创建一个三维优化空间来表示资源分配配置,而后通过搜索该空间为应用生成多种并行机制,最后对各种并行机制进行静态评估从而找出最优的机制。该方法同时考虑了任务并行、数据并行以及通讯开销。在异构多核处理器 Cell 上利用一个图像处理应用对此方法进行了测试。实验表明,这种模型驱动的方法能够很好地评估性能并为应用确定有效的并行机制。

**关键词** 多核处理器,资源分配模型,BSP 模型,任务并行,数据并行

## 0 引 言

在单芯片上集成多个处理器核已成为当前处理器设计的主流。STI(SONY、TOSHIBA 和 IBM)联合开发的 Cell 宽带引擎(Cell Broadband Engine, Cell BE)[1]是多核处理器的代表,也是目前学术界最为关注的异构多核处理器。Cell BE 包含一个 PowerPC 核(power 处理器(power processing element, PPE))和 8 个向量协处理器(synergistic processing elements, SPE),PPE 运行操作系统,SPE 执行计算密集型任务。每个 SPE 拥有 256KB 本地存储,SPE 通过内部总线与主存或其它 SPE 进行数据传输。近期的研究[2,3]证实 Cell BE 具有很强的计算能力。

然而,如何利用 Cell 的并行计算能力给程序员带来了很大的挑战。首先,多核上的计算资源和存储资源需要进行显示管理。其次,应用程序的并行计算部分需要分配到 SPE 上进行运算,如何充分开发多层次并行,同时减少通讯,仍是并行程序设计人员需要考虑的问题。针对以上问题,众多科研机构及生产商致力于为 Cell 开发新的编程模型和软件编程环境来减轻编程负担。Bellens 等[4]提出了 CellSs 用于开发程序的任务并行,CellSs 基于一个源到源 C 编译器和一个运行时系统,运行时系统负责任务及数据的调度。Blagojevic 等[5]提出了 MMGP 模型来处理多核处理器的多级并行,然而 MMGP 没有考虑数据局部性,同时也不能准确地估算通讯代价。其它工作[3,6]主要关注多核代码自动生成技术,包括开发 Cell 的线程级并行及单指令多数据(single instruction multiple data, SIMD)并行。

与以上工作不同,本文提出了一个资源分配模型(resource allocation model, RSA),该模型综合考虑了多级并行及通讯代价。RSA 主要基于整体同步并行(bulk synchronous parallel, BSP)模型[7],在 RSA 中,一个多任务程序被抽象为一系列的计算步和通讯步,每个计算步集合了可以并行执行的任务,每个通讯步传输相邻计算步间的数据。每个任务的数据并行由一个源到源编译器开发[8]。为了获得多任务程序的最优并行策略,RSA 建立了一个三维优化空间,并通过搜索该空间来为程序得到高效的并行机制。我们在 Sony 发布的基于 Cell 处理器的 PlayStation 3 平台上,利用 RSA 对图像处理应用 Sobel 进行测试。实验表明,RSA 能很好地进行性能评估,并能为应用确定最优的并行机制。

## 1 问题陈述

给定一个多任务并行程序 P,程序中的每个任务对应于一个数据并行的计算密集型函数。程序 P 可以被抽象为一个有向无环图(directed acyclic graph, DAG),又称任务图,记为 $G(V,E)$。其中,结点 $v \in V$ 表示 P 中的一个任务,边 $(u,v) \in E$ 表示

① 973 计划(2007CB310900)资助项目。
② 女,1981 年生,博士生;研究方向:多核编译及代码生成。
③ 通讯作者,E-mail:wangmiao@nudt.edu.cn
(收稿日期:2009-08-21)

任务 $u$ 与任务 $v$ 之间存在数据依赖关系，即任务 $v$ 使用由任务 $u$ 定义的数据。基于 BSP 模型，我们将程序 P 的任务图 G 映射到 Cell BE 上，以下四个方面因素会影响并行策略的选择：

- 根据 Amdahl 定律[9]，当并行化一个任务时，性能提升是受限的；
- 需要充分利用计算资源来保持并行任务间的负载平衡；
- 为单个任务分配更多的计算资源会限制可并行执行的任务数；
- 数据并行会减少可利用的存储资源，也会引入额外的通讯。

考虑到以上几方面，为一个多任务应用确定最优的多级并行策略并不简单。图 1 显示了一个三任务程序的任务图。图 2 显示了基于 BSP 模型将图 1 中任务图映射到 Cell 上的两种并行策略。图 2 中的 comp 表示 BSP 模型中的计算步，comm 表示通讯步，虚线箭头表示数据重用，实线箭头表示数据传输。在图 2(a)的并行策略中，3 个任务对应 3 个计算步，每个任务中的数据被分布到 4 个 SPE 上，相邻计算步间进行数据重用。在图 2(b)的并行策略中，任务 $task_0$ 和 $task_1$ 对应第 1 个计算步，每个任务中的数据分别被分配到 2 个 SPE 中，任务 $task_2$ 对应第 2 个计算步，两个计算步之间通过通讯进行数据重新分布。图 2(a)的并行策略最大化地利用了数据并行，图 2(b)中的策略同时开发了任务并行和数据并行，但却引入了通讯开销。为了确定多任务应用的有效并行机制，我们基于 Cell 建立了一个资源分配模型，该模型考虑了单任务的数据级并行以及多任务应用的任务级并行，能够很好地为应用确定最优的并行策略。

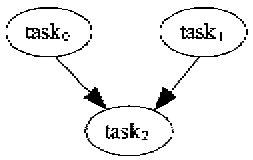


图 1 任务图

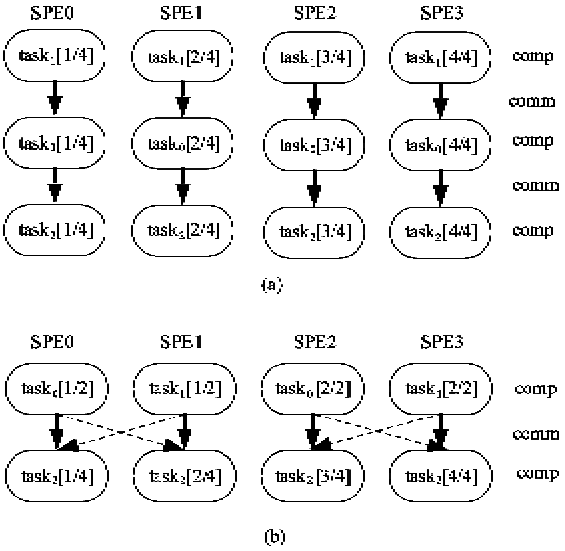


图 2 两种并行策略：(a)策略一；(b)策略二

## 2 单任务数据级并行开发

给定一个任务，我们利用一个源到源编译器[8]来开发任务的数据级并行，为任务生成单程序多数据(single program multiple data, SPMD)并行代码。首先，编译器利用仿射图[10]及仿射图划分来计算程序中数组维数间的对齐关系。其次，根据数据对齐结果，编译器对程序中的数组进行按块划分，并将划分后的数组分配到不同 SPE 的局部存储中。最后，编译器为程序插入通讯(如，shift 操作)，使得每个 SPE 能够通过远程访问获得其计算所需但在其他 SPE 上的数据。编译器能够根据不同数据划分以及不同 SPE 数，为单任务生成不同版本的 SPMD 代码。对于任务的一个 SPMD 代码版本，我们称之为一个变元(variant)。图 3 显示了单任务 code 的变元生成过程。我们利用 $< code, DDS_i, j\ SPE >$ 表示任务 code 的一个变元，即任务 code 在数据划分机制(data distribution scheme)为 $DDS_i$，$SPE$ 数为 $j$ 时的一个 SPMD 代码版本。一个变元具有以下几个属性：

- 执行时间；
- 每个 SPE 的存储访问空间；
- 数据划分机制及所使用的 SPE 数。

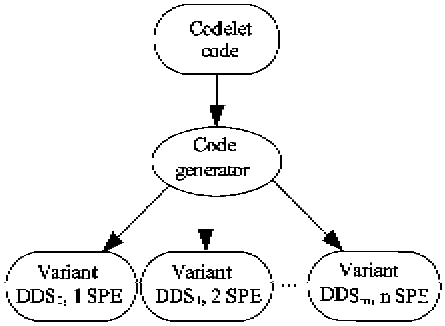


图 3 变元生成

## 3 多任务多级并行开发

为了有效地分配多核的存储资源和计算资源，我们为多任务程序创建了一个资源分配模型 RSA。

该模型首先构建了一个三维的优化空间,第一维是变元选择,用于决定数据并行;第二维是任务分组,用于决定任务并行;第三维是处理单元(processing element, PE)分配,用于确定数据通讯。基于优化的搜索空间,RSA 评估和寻找最优的并行策略。

### 3.1　变元选择

给定一个任务,编译器[8]为任务生成多个变元。因此,对于一个多任务程序,每个任务将对应多个变元。变元选择就是为程序中的每个任务选择一个变元。RSA 通过枚举来计算所有任务的所有变元的组合。图 4 显示了一个变元选择的简单例子。对于图 1中的 3 个任务 $task_0$, $task_1$ 和 $task_2$,每个任务在图 4中都有 3 个变元,例如 $task_0$,其变元分别为 $< task_0, DDS_0, 1\ SPE >$, $< task_0, DDS_1, 2\ SPE >$ 和 $< task_0, DDS_2, 4\ SPE >$。令 $VC_{i,j,k}$代表 3 个任务的某种变元选择方式,如图 4 所示,$VC_{0,0,1}$表示 3 个任务的一种变元选择,即任务 $task_0$, $task_1$ 和 $task_2$ 分别选择了变元 $< task_0, DDS_0, 1\ SPE >$, $< task_1, DDS_0, 1\ SPE >$ 和 $< task_2, DDS_1, 2\ SPE >$。图 4 中 3 个任务共有 $3 \times 3 \times 3 = 27$ 种不同的变元选择。

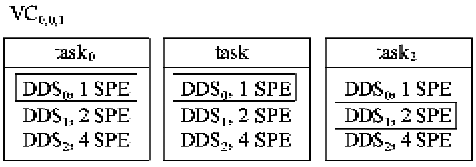


**图 4　变元选择**

### 3.2　任务分组

任务分组用于确定一个多任务程序中的哪些任务可以并行执行。给定程序的任务图 G,对于 G 中的任务 task,当 task 的所有前驱任务均已执行之后,task 才能够被调度。根据从 G 根结点(即没有前驱的结点)到任务 task 的距离,我们为 task 分配一个序号(rank,rank≥1),用以表示任务的执行顺序。任务分组时,序号相同的任务将被划分到同一个子集或组中,同一组中的任务可以并行执行。我们根据序号为各组进行排序,并按照序号从低到高将各组中的任务分配到不同的计算步中。

图 5 显示了对图 1 中的任务图进行任务分组的一个例子。对于图 1 中的 3 个任务,若 3 个任务顺序执行,则程序完成需要的最大计算步数为 3。模型 RSA 首先为图 5 任务图中的每个任务确定序号范围。例如,对于任务 $task_0$,若利用 ASAP(as soon as possible)调度,$task_0$ 获得最小序号值 1;若利用 ALAP(as late as possible)调度,$task_0$ 获得最大序号值 2。因此,任务 $task_0$ 的序号范围是[1,2]。同样,任务 $task_1$ 和 $task_2$ 的序号范围分别为[1,2]和[2,3]。根据每个任务的序号范围,RSA 枚举所有任务的所有可能序号值。图 5 显示了枚举后 3 种可能的分组情况,$G_i$ 表示一个分组策略。在分组策略 $G_1$ 中,任务 $task_0$ 和 $task_1$ 的序号取值为 1,任务 $task_2$ 的序号取值为 2,因此 $task_0$ 和 $task_1$ 被划分到同一个组 $g_1$ 中,$task_2$ 被划分到另一个组 $g_2$ 中。根据 $g_1$ 和 $g_2$ 的序号顺序,RSA 将 $g_1$ 中的两个并行任务 $task_0$ 和 $task_1$ 分配到第一个计算步中,将 $g_2$ 中的任务 $task_2$ 分配到第二个计算步中。在任务分组时,RSA 将对每一个分组策略进行有效性检测,进而排除无效策略,如图 5 中 $G_2$。

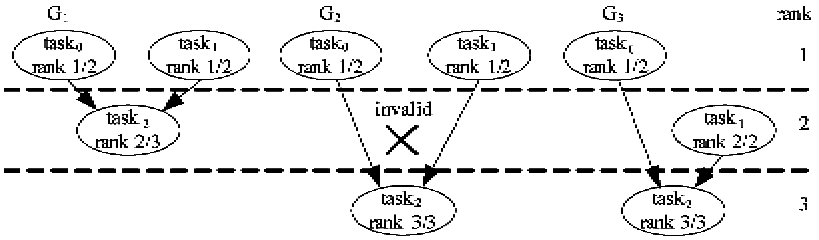


**图 5　任务分组策略**

### 3.3　PE 分配

在变元选择和任务分组之后,PE 分配的搜索空间将被建立。对于一个多任务程序,给定一个任务分组策略 $G_i$ 和一种变元选择方式,PE 分配将为 $G_i$ 每个组中的每个任务分配不同的处理器单元。令 $g$ 表示 $G_i$ 中的一个组,且 $g$ 由 $m$ 个任务 $task_1, task_2, \cdots, task_m$组成。根据变元选择,若任务 $task_i (1 \leqslant i \leqslant m)$对应的变元为 $< task_i, DDS_i, j_iSPE >$,则 $g$ 中 $m$ 个任务所需的 SPE 数 $N_g = j_1 + j_2 + \cdots + j_m$。令 $N_{pe}$ 表示可获得的 SPE 数,则 PE 分配时,需要保证 $N_g \leqslant N_{pe}$。PE 分配后,每个 SPE 中的数据和代码量不能超过 SPE 的存储容量。图 6 显示了变元选择为

图4中 $VC_{0,0,1}$，分组策略为图5中 $G_1$ 时 PE 分配的结果。由 $VC_{0,0,1}$ 可知，任务 $task_0$ 和 $task_1$ 根据数据划分机制 $DDS_0$ 分布到1个 SPE 上，任务 vgrad 根据机制 $DDS_1$ 被划分到2个 SPE 上。而且，$G_1$ 将3个任务分成两组分配到两个计算步中。因此，在图6的第一个计算步中，每个任务分别分配一个 SPE，在第二个计算步中，任务 $task_2$ 中的数据被分解到2个 SPE 中。PE 分配后，两个计算步之间将加入一个通讯步进行数据传输(图6中实线箭头所示)及数据重用(图6中虚线箭头所示)。

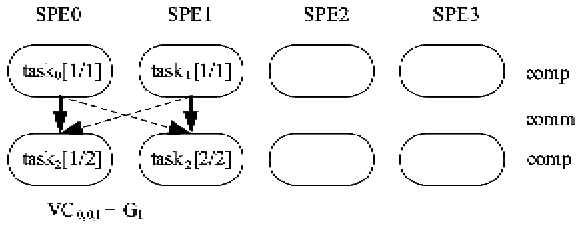


**图6 PE 分配**

### 3.4 并行策略评估

变元选择、任务分组和 PE 分配为多任务程序构成了一个三维的搜索空间，该空间的每个元素对应于一个并行策略。通过遍历空间每一维上的不同配置，资源分配模型 RSA 为应用生成多种并行策略。图6和图7分别显示了图1任务图对应的搜索空间中两个不同的并行策略，图6中的策略包含了2个计算步和1个通讯步，而图7中的策略则含有3个计算步和2个通讯步。为了判断不同并行策略的优劣，RSA 模型利用 BSP 模型中的代价函数评估每个策略的性能，并从中选出最优的。

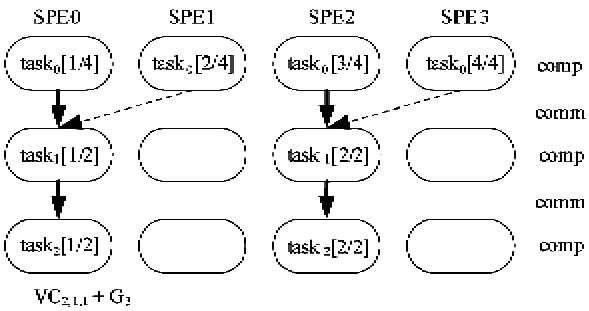


**图7 并行策略**

给定一个多任务程序 P 和其搜索空间的一个候选并行策略 S，假设 S 由 $N_{comp}$ 个计算步和 $N_{comm}$ 个通讯步组成。根据 BSP 代价模型，对于 S 中的任意计算步 p，该计算步对应于一个含有 $n$ 个并行任务的组。令 $T_i$ 表示任务 $i(1 \leqslant i \leqslant n)$ 所选变元的执行时间，则计算步 p 的计算代价 $T_{comp}$ 为

$$T_{comp}(p) = \max_{1 \leqslant i \leqslant n}\{T_i\}$$

对于 S 中任意的通讯步 q，通讯代价 $T_{comm}$ 为：

$$T_{comm}(q) = T_{sync}(q) + T_{fetch}(q) + T_{pe}(q) + T_{put}(q)$$

其中，$T_{sync}$ 为同步代价，$T_{feeth}$ 为从主存中取数的代价，$T_{pe}$ 为 PE 间通讯代价，$T_{put}$ 为将数据放回主存的代价。

并行策略 S 的总执行时间 $T_{total}$ 为所有计算步和通讯步的代价之和，即

$$T_{total} = \sum_{p \in N_{comp}} T_{comp}(p) + \sum_{q \in N_{comm}} T_{comm}(q)$$

通过比较不同策略的执行时间，RSA 模型可以为程序确定最有效的并行策略。

## 4 实 验

本部分我们将利用 RSA 模型对 Sobel 边缘检测算法进行测试与性能评估。

### 4.1 实验框架

我们利用3个工具为多任务程序生成和评估并行策略：

- 程序分析器用于提取多任务程序中的单个任务并构建任务图。
- 源到源编译器处理程序中的每个单任务，并为其生成多个变元。每个变元对应一个 PPE 的 C 文件、一个 SPE 的 SPMD C 文件和一个描述数据划分机制的文件。变元的执行时间通过一个模版程序进行统计。
- 资源分配模型 RSA 建立搜索空间并确定有效的并行机制。

实验测试平台是 Sony 的 PlayStation 3，PlayStation 3 有6个可用的 SPE。模型生成的并行策略通过手动实现。我们利用 IBM SDK Cell 3.0 和 GNU C 编译器 gcc 4.1 对 PPE 和 SPE 的 C 程序进行编译，生成二进制代码。

### 4.2 Sobel 应用

边缘检测是勾绘图像中物体轮廓的计算机视图的常用方法之一，Sobel 边缘检测[11]使用 3×3 大小的窗口来计算水平和垂直方向的梯度。图8显示了 Sobel 的任务提取和任务图。我们从 Sobel 中提取了七个任务：left、right、down、up、sub、magnitude 和 direction。图8的任务图说明了任务之间的相关性。我们把图8中任务分成3类：

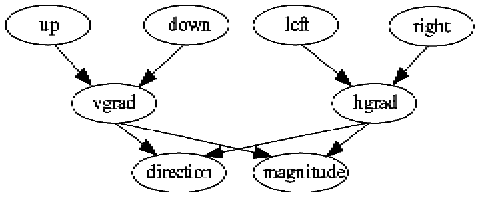


**图 8　Sobel 中的任务和任务图**

• left,right,down 和 up 计算梯度的中间值,他们需要对输入数据进行 shift 操作,从而可以利用 SIMD 和数据并行;

• vgrad 和 hgrad 各自表示 y 和 x 方向的计算,执行矩阵相减操作;

• magnitude 和 direction 使用梯度值来计算各边的方向和大小,都使用了浮点计算。

对于每个任务,我们的源到源编译器为其生成 5 个不同的变元,分别对应于不同的 SPE 个数和不同的数据划分机制。

### 4.3　实验结果

我们使用资源分配模型 RSA 处理 Sobel 程序。首先,RSA 搜索变元选择和任务分组这两个维度,使用简单的 PE 分配机制将会得到几百万个有效的并行策略。对于 64×64 的输入,通信代价的平均值为 547 个基准时间,计算代价平均为 26900 个基准时间。若使用最坏估计作为上限,大约有 2000 个变元选择方式的计算代价在上限之内,误差仅有 5%。对于程序员来说实现这么多的代码是件十分困难的事。我们的模型 RSA 能够有助于挑选出一些最有效的代码。

然后,RSA 遍历任务分组和 PE 分配两个维度,3 个最好的并行机制被选择了出来,它们包括:

• gamma:为每个梯度运算分配两个 SPE,这时需要 shift 操作实现 SPE 间的数据通讯。梯度运算的结果再分配到 4 个 SPE 上供 magnitude 和 direction 进行计算;

• beta:使用 4 个 SPE 同时计算 up,down,left 和 right。每个 SPE 共享计算结果,vgrad,hgrad,magnitude 和 direction 也分配在 4 个 SPE 上;

• alpha:分别分配所有的计算在 4 个 SPE 上,这时需要全局 shift 操作来进行梯度的计算,但无需数据传输。

为了比较,我们同时实现了另外 3 个代码,分别是:单 PPE 上执行的双线程代码、单 SPE 代码 mono 和双 SPE 代码 duo。

图 9 为相对于 PPE 的各种实现的加速比。从图 9可知,使用 4 个 SPE 能够带来最好的加速比,与单个 SPE 相比,alpha,beta 和 gamma 分别为 3.91、3.96 和 3.88 倍。由此可以看出,RSA 模型能够为应用选择出最好的几种并行机制,同时也极大地减轻了程序员的负担。

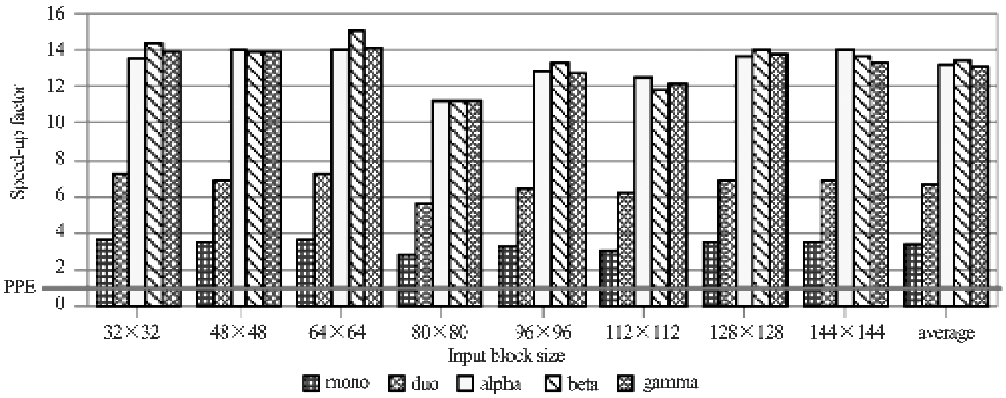


**图 9　性能比较**

## 5　结　论

本文基于 Cell BE 实现了一个资源分配模型 RSA,该模型构建了一个三维优化空间,空间的 3 个维度分别包括变元选择、任务分组以及 PE 分配。RSA 通过搜索空间中每一维的配置,为多任务应用程序在多核上的实现提供有效的并行机制。模型同时考虑了数据并行和任务并行,以及任务间的通讯开销,很好地开发了程序的数据局部性及各种粒度的并行。我们利用图像处理应用 Sobel 对 RSA 模型进行了测试,实验表明,RSA 模型能够为应用选择最

优的并行策略,并且能够极大地减轻程序员的编程负担。在下一步的工作中,我们将分析更复杂的通讯模式,进一步精确通讯代价评估,并实现支持多任务自动并行的代码生成器。

**参考文献**


[1] Gschwind M, Hofstee H, Flachs B, et al. Synergistic processing in cell's multicore architecture. *IEEE MICRO*, 2006, 26(2):10-24

[2] Petrini F, Fossum G, Fernandez J, et al. Multicore Suprises: Lesson Learned from Optimizing. Sweep3D on the Cell Broadbend Engine. In: Proceedings of IPDPS 2007, Long Beach, USA, 2007. 26-30

[3] Eichenberger A, O'Brien J, O'Brien K, et al. Using advanced compiler technology to exploit the performance of the cell broadband engine architecture. *IBM Systems Journal*, 2006, 45(1): 59-84

[4] Bellens P, Perez J, Badia R, et al. CellSs: a programming model for the Cell BE architecture. In: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, Tampa, USA, 2006. 5-5

[5] Blagojevic F, Feng X, Cameron D, et al. Modeling multi-grain parallelism on heterogeneous multi-core processors: a case study of the Cell BE. In: Proceedings of the 2008 International Conference on High-Performance Embedded Architectures and Compilers, Sweden, 2008. 38-52

[6] Zhao Y, Kennedy K. Dependence-Based code generation for a Cell processor. In: Proceedings of the 19th International Workshop on Languages and Compilers for Parallel Computing, New Orleans, USA, 2006. 64-79

[7] Valiant L. A bridging model for parallel computation. *Communications of the ACM*, 1990, 33(8):103-111

[8] Wang M, Bodin F, Matz S. Automatic data distribution for improving data locality on the Cell BE architecture. In: Proceedings of the 22nd International Workshop on Languages and Compilers for Parallel Computing, Newark, USA, 2009.

[9] Amdahl G. Validity of the single processor approach to achieving large scale computing capabilities. In: AFIPS Conference Proceedings, 1967, 30(8): 483-485

[10] Li J, Chen M. The data alignment phase in compiling programs for distributed-memory machines. *Journal of Parallel and Distributed Computing*, 1991, 13(2): 213-221

[11] Sobel I. An isotropic 3x3 image gradient operator. *Machine Vision for Three-Dimensional Scenes*, 1990, 376-379


# A resource allocation model for heterogeneous multi-core Cell


Wang Miao, Wang Zhiying, Wu Guiming

(Department of Computer Science, National University of Defense Technology, Changsha 410073)



**Abstract**

To fully utilize the multi-level parallelism delivered by heterogeneous multi-core processors and solve the problem of multi-core resource allocation, this paper presents a model-driven technique for mapping multi-task parallel programs onto multi-core platforms. When using the technique, the resource allocation configurations are expressed within a three-dimensional optimization space which affects the task-level parallelism and the data-level parallelism, and communication. Then, the performance of each valid parallelization scheme is estimated statically, taking both the computation cost and the communication cost into account. The experiment on the approach on the Cell BE using an image processing application. The experimental results show that the proposed model-driven method can correctly predict the overall performance and highlight the most efficient parallelization schemes.



**Key words**: multi-core processor, resource allocation model, BSP model, task parallelism, data parallelism