软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2011,22(9):2222−2234 [doi: 10.3724/SP.J.1001.2011.03883]

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

异构系统功耗感知的并行循环调度方法*

王桂彬+, 杨学军, 徐新海, 林一松, 李 鑫

(国防科学技术大学 计算机学院 并行与分布处理国家重点实验室,湖南 长沙 410073)

Power-Aware Parallel Loop Scheduling Method for Heterogeneous System

WANG Gui-Bin+, YANG Xue-Jun, XU Xin-Hai, LIN Yi-Song, LI Xin

(National Key Laboratory of Parallel and Distributed Processing, College of Computer, National University of Defense Technology, Changsha 410073, China)

+ Corresponding author: E-mail: wgbljl@gmail.com

Wang GB, Yang XJ, Xu XH, Lin YS, Li X. Power-Aware parallel loop scheduling method for heterogeneous system. *Journal of Software*, 2011,22(9):2222−2234. http://www.jos.org.cn/1000-9825/3883.htm

Abstract: Based on the OpenMP-like parallel program, a loop scheduling and dynamic voltage scaling technology is coordinated to optimize system power consumption under the given performance constraint. First, the basic model for power-aware loop scheduling on the heterogeneous system is presented. After that, through theoretical analysis, it has been concluded that the lower bound of energy consumption for parallel loop scheduling on heterogeneous systems, can be used as a baseline to evaluate the efficiency of optimization technology. Furthermore, this paper induces the scheduling problem as a typical integer programming problem and proposes inner-processor loop re-scheduling method to further reduce power consumption. Finally, 10 typical kernel programs on a CPU-GPU heterogeneous system are created. The experimental results demonstrate that the proposed method can effectively reduce the total energy consumption of the whole system and improve the system energy efficiency.

Key words: power optimization; heterogeneous system; loop scheduling; OpenMP

摘 要: 以类 OpenMP 的并行程序为研究对象,在满足性能约束的条件下,结合异构系统并行循环调度和处理器动态电压调节技术优化系统功耗.首先建立了异构系统功耗感知的并行循环调度问题基本模型;然后,通过分析方法给出异构系统并行循环调度的能耗下界,该下界可用于评估功耗优化方法的实际效率;进而将异构系统并行循环调度问题归纳为整数规划问题,在此基础上,提出了处理器内循环再调度方法进一步降低功耗.最后,以 CPU-GPU 异构系统为平台评测了 10 个典型 kernel 程序.实验结果表明,该方法可以有效降低系统功耗,提高系统效能.

关键词: 功耗优化;异构系统;循环调度;OpenMP

中图法分类号: TP316 **文献标识码**: A

随着系统规模的不断增大,高性能计算机系统的功耗问题日趋严峻.以 2009 年 11 月发布的 Top500[1]超级计算机排名之首的 Cray“美洲豹”为例,系统功耗已接近 7MW.因此,高性能计算机系统的功耗优化问题是当前国际上研究热点之一.

* 基金项目: 国家自然科学基金(60921062, 60903059, 60903044); 国家科技重大专项(2009ZX01036-001-003-001)
收稿时间: 2010-03-19; 定稿时间: 2010-04-28

异构系统已成为高性能计算机的发展趋势,Top500 中排名第 2 的 IBM 超级计算机“走鹃”就是异构系统的典型案例.该系统由通用 CPU 和 CELL 加速器构成,单位功耗的计算性能(效能)达到 444.25MFlops/w,在 Green 500[2]排名中位列第 6.异构高性能计算机系统在具备高峰值计算性能的同时,也给系统功耗优化带来了新的技术挑战.由于异构系统中不同类型的计算资源具有不同的计算能力和实际效能,因此,已经提出的面向同构并行系统的低功耗优化技术,如文献[3−5],均不适用于异构系统.

目前,在学术领域面向异构系统的功耗研究还处于起步阶段.我们认为,功耗感知的并行循环调度方法是异构系统功耗优化的关键技术之一.该技术旨在将并行循环划分并映射在异构处理器上执行,结合处理器动态电压/频率调节技术(DVFS)优化系统功耗.文献[3]面向同构系统分析了任务调度算法对系统能耗的影响,本文将该理论结果扩展到异构系统,并提出了异构系统功耗感知的并行循环调度方法.本文创新点主要包括:

(1) 建立功耗感知的异构系统并行循环调度模型,并通过分析方法得出时间约束下异构系统并行循环调度的能耗下界;

(2) 将异构系统并行循环调度问题归纳为整数规划问题,并给出降低计算复杂度的优化方法;针对处理器频率离散可调的现状,提出处理器内循环再调度方法以进一步优化功耗;

(3) 我们以 CPU-GPU 异构系统为实验平台,通过 10 个典型的应用程序验证了本文提出的并行循环调度方法可以有效降低系统能耗,提高系统效能.

本文第 1 节给出异构系统并行循环调度问题的基本模型和概念.第 2 节讨论异构系统并行循环调度方法及优化策略.第 3 节介绍 CPU-GPU 异构系统软硬件实验平台,并给出实验评测与分析.第 4 节介绍目前国内外功耗优化的相关研究进展.第 5 节总结本文工作,并给出今后有待继续研究的问题.

1 模型与概念

本文以类 OpenMP 的并行程序为研究对象,主要研究并行程序中并行循环面向异构系统的循环调度和功耗优化的关系.

图 1 是一个典型的并行循环——doall 循环,该循环不存在跨迭代的数据依赖,因此可以并行执行.设 doall 循环的迭代数为 N,并假设其在包含 m 个不同能力处理器的异构系统上并行执行.

```
doall i=1, N
    Block
end doall
```

Fig.1 A DOALL loop sample

图 1 DOALL 循环示例

不失一般性,面向异构系统的并行循环调度问题可以描述为:将 doall 的迭代空间进行划分,并映射到异构处理器上,记为

$$F_{doall}=\langle I_1,I_2,\ldots,I_m\rangle,$$

其中,$I_j(1\leqslant j\leqslant m)$是 doall 循环迭代集合的子集,且被映射到第 j 个处理器上.N_j 表示子集 I_j 的大小,即迭代次数;T_j 和 E_j 分别表示第 j 个处理器完成迭代子集 I_j 的执行时间和能量消耗.

那么,该循环在此异构系统上的执行时间为

$$T_{paral}=\max_{1\leqslant j\leqslant m}T_j,$$

而总能耗为

$$E_{paral}=\sum_{1\leqslant j\leqslant m}E_j.$$

按照物理学定律,能耗 E 是处理器功耗 P 与执行时间 T 的乘积,即 $E=P\times T$.下面分析处理器功耗与执行时间的关系,首先定义 S 为处理器的执行速度,即单位时间内完成的迭代次数:

$$S=N/T \tag{1}$$

根据 CMOS 电路的功耗公式,处理器动态功耗与电压和频率的关系为

$$P=ACV^2f \tag{2}$$

其中,A 是切换因子,C 是切换电容,V 是核心电压,f 是运行频率.

处理器核心电压与该电压下的最高运行频率有如下关系:

$$f_{\max}=K\frac{(V-V_T)^{\gamma}}{V}(1\leqslant\gamma\leqslant 2),$$

其中,V_T 为阈值电压,K 和 γ 是与工艺相关的参数.一般 V_T 很小,因此上式简化为

$$f_{\max}=KV^{\gamma-1} \tag{3}$$

在本文中,我们假设处理器工作在核心电压允许的最高频率,因此在后文中省略下标 max.

为了分离出运行频率的因素,记 $X=S/f$,因此 X 与处理器频率无关.综合公式(2)和公式(3)有

$$S=Xf=X(K^{2/(\gamma-1)}P/(AC))^{(\gamma-1)/(\gamma+1)}.$$

为便于描述,令,$\alpha=(\gamma+1)/(\gamma-1)$,$\mu=X(K^{2/(\gamma-1)}/(AC))^{(\gamma-1)/(\gamma+1)}=(XK^{1-1/\alpha})/(AC)^{1/\alpha}$,可得

$$S=\mu P^{1/\alpha} \tag{4}$$

需要注意的是,公式(1)~公式(4)是对所有处理器的一般性描述,因此省略了处理器索引下标.

在满足时间约束 T 的条件下,异构系统功耗感知的并行循环调度问题可表示为

$$\begin{cases}\min\ E=\sum\limits_{1\leqslant j\leqslant m}P_j\times T_j\\ \text{s.t.}\ \max\limits_{1\leqslant j\leqslant m}T_j\leqslant T\\ \sum\limits_{1\leqslant j\leqslant m}N_j=N\\ N_j\geqslant 0,\ N_j\in Z\end{cases} \tag{5}$$

根据上述公式(1)~公式(5),本文采用动态电压/频率调节,结合处理器循环调度来优化系统总能耗.

2 异构系统并行循环调度方法

从公式(5)可以看出,循环划分结果和处理器运行频率都会影响系统总能耗;而处理器的运行频率与其完成的循环迭代量存在一定关系.因此,公式(5)从原理上可以分为两步求解.第 2.1 节首先讨论在循环调度给定的情况下,即 $F_{doall}=\langle I_1,I_2,\ldots,I_m\rangle$,确定处理器最优运行频率.基于该分析,将公式(5)描述的并行循环调度问题归纳为整数规划问题.第 2.2 节基于对该整数规划问题的分析,得出异构系统并行循环调度的能耗下界.第 2.3 节给出了并行循环调度问题的优化求解方法.同时,针对处理器频率离散可调的现状,提出处理器内循环再调度方法以进一步优化功耗.

2.1 给定循环调度的处理器最优频率求解

根据上一节的陈述,在第 $j(1\leqslant j\leqslant m)$个处理器上执行迭代集合 I_j 的能耗与处理器运行频率有关.因此,我们首先讨论在给定循环划分的情况下,处理器最优频率选择问题.不失一般性,设第 j 个处理器在完成迭代集合 I_j 的过程中,每次迭代都可以运行在不同的频率下,记为 $\langle f_j^1,f_j^2,\ldots,f_j^{N_j}\rangle$,相应的执行速度和功耗分别记为 $\langle S_j^1,S_j^2,\ldots,S_j^{N_j}\rangle$ 和 $\langle P_j^1,P_j^2,\ldots,P_j^{N_j}\rangle$.那么,完成迭代集合 I_j 的总执行时间 $T_j=\sum\limits_{1\leqslant k\leqslant N_j}1/S_j^k=\sum\limits_{1\leqslant k\leqslant N_j}1/(X_jf_j^k)$,而总能耗为 $E_j=\sum\limits_{1\leqslant k\leqslant N_j}(S_j^k)^{\alpha-1}/\mu_j^{\alpha}=\sum\limits_{1\leqslant k\leqslant N_j}(X_jf_j^k)^{\alpha-1}/\mu_j^{\alpha}$ (由公式(4)可得).因此,处理器最优频率选择问题可以描述为

$$\begin{cases}\min\ E_j=\sum\limits_{1\leqslant k\leqslant N_j}(X_jf_j^k)^{\alpha-1}/\mu_j^{\alpha}\\ \text{s.t.}\ \sum\limits_{1\leqslant k\leqslant N_j}1/(X_jf_j^k)\leqslant T\end{cases} \tag{6}$$

假设处理器频率连续可调,定理 1 给出该问题的最优解.

定理 1. 对于第 j 个处理器,在时间约束 T 的条件下完成 N_j 次迭代的能耗最优解是所有迭代的计算过程都在相同的频率下完成,此时功耗 $P_j^k=N_j^\alpha/(\mu_j^\alpha T^\alpha)$ $(1\leqslant k\leqslant N_j)$,能耗最优值 $E_j^*=N_j^\alpha/(\mu_j^\alpha T^{\alpha-1})$.

证明:由公式(6)可知,E_j 可以看作是关于 f_j^k $(1\leqslant k\leqslant N_j)$的 N_j 元函数.不难想象,由于假设处理器频率连续可调,因此能耗最优值必然出现在总执行时间恰好等于约束时间 T 的情况下.因此,约束条件 F 为等式:

$$\sum_{1\leqslant k\leqslant N_j}1/(X_j f_j^k)-T=0.$$

由公式(4),上式可以转化为

$$\sum_{1\leqslant k\leqslant N_j}1/(\mu_j(P_j^k)^{1/\alpha})-T=0.$$

同理,由公式(4),E_j 可以转化为

$$E_j=\sum_{1\leqslant k\leqslant N_j}(P_j^k)^{(\alpha-1)/\alpha}/\mu_j.$$

下面通过 Lagrange 乘子法求解能耗最优值.设λ为 Lagrange 乘子,则有$\nabla E=\lambda\nabla F$,即

$$\frac{\partial E_j}{\partial P_j^k}=\lambda\frac{\partial F}{\partial P_j^k}\ (1\leqslant k\leqslant N_j).$$

可得 $\frac{\alpha-1}{\mu_j\alpha}(P_j^k)^{-1/\alpha}=\frac{\lambda}{\mu_j}\left(-\frac{1}{\alpha}\right)(P_j^k)^{-(\alpha+1)/\alpha}$,即 $P_j^k=\frac{\lambda}{1-\alpha}$. 带入约束条件 F 可得 $\lambda=\frac{(1-\alpha)N_j^\alpha}{\mu_j^\alpha T^\alpha}$, 故 $P_j^k=\frac{N_j^\alpha}{\mu_j^\alpha T^\alpha}$.可以看出,当所有迭代过程都在相同的功耗下计算,则完成 N_j 次迭代的总能耗最优.因此,第 j 个处理器完成 N_j 个迭代的最优能耗为 $E_j^*=\frac{N_j^\alpha}{\mu_j^\alpha T^{\alpha-1}}$. □

定理 1 讨论单处理器能耗最优的频率选择问题.基于该结论,定理 2 给出异构多处理器系统总能耗与各处理器迭代数量的关系.

定理 2. 对于问题将并行循环 L 的 N 个迭代分配在 m 个异构处理器,在给定循环划分$\langle I_1,I_2,\ldots,I_m\rangle$的情况下,当第 j 个处理器的功耗为 $P_j=N_j^\alpha/(\mu_j^\alpha T^\alpha)$ $(1\leqslant k\leqslant m)$,完成循环 L 的总能耗最小,系统总能耗最优值为

$$E^*=\left(\sum_{1\leqslant j\leqslant m}\frac{N_j^\alpha}{\mu_j^\alpha}\right)\Big/T^{\alpha-1}.$$

证明:在给定循环划分$\langle I_1,I_2,\ldots,I_m\rangle$的情况下,系统总能耗最优要求每个处理器的能耗都达到最优.由定理 1 可知,对于第 j 个处理器,当处理器以恒定功率 $N_j^\alpha/(\mu_j^\alpha T^\alpha)$ 完成迭代子集 I_j 时,可以在满足时间约束 T 的条件下达到能耗最优,此时,能耗最优值 $E_j^*=N_j^\alpha/(\mu_j^\alpha T^{\alpha-1})$.因此,对于由 m 个处理器构成的系统而言,总能耗最优值 $E^*=\sum_{1\leqslant j\leqslant m}E_j^*=\left(\sum_{1\leqslant j\leqslant m}\frac{N_j^\alpha}{\mu_j^\alpha}\right)\Big/T^{\alpha-1}$. □

并行循环调度问题要求满足时间约束,即$\forall j\in[1,m]$,满足 $T_j\leqslant T$;由公式(1)可知,$T_j=\frac{N_j}{S_j}=\frac{N_j}{X_j f_j}$.因此,$N_j$ 应满足不等式 $N_j\leqslant f_jX_jT$.记第 j 个处理器的最高频率为 F_j,可得

$$N_j\leqslant F_jX_jT \tag{7}$$

根据定理 2 和公式(7),公式(5)可以进一步描述为

$$\begin{cases}\min\ E=\left(\sum\limits_{1\leqslant j\leqslant m}\dfrac{N_j^{\alpha}}{\mu_j^{\alpha}}\right)\Big/T^{\alpha-1}\\ \text{s.t.}\ \ N_j\leqslant F_jX_jT,\ 1\leqslant j\leqslant m\\ \quad\ \sum\limits_{1\leqslant j\leqslant m}N_j=N\\ \quad\ N_j\geqslant 0,\ N_j\in Z\end{cases}\tag{8}$$

可以看出,公式(8)属于整数规划问题.通过整数规划算法可以求得最优划分 $F_{doall}=\langle I_1,I_2,\ldots,I_m\rangle$(考虑局部性因素,我们令 I_j 为循环 L 中连续的 N_j 个迭代),然后由定理 1 确定处理器最优频率.对比公式(5)和公式(8)可以发现,前者包含 $2m$ 个自变量,而后者仅包含 m 个自变量,计算复杂度明显降低.

2.2 并行循环调度能耗下界分析

与同构系统相比,异构系统为提高系统效能提供了更大的优化空间[6],但是尚未有研究对其进行定量分析.因此,本节以并行循环为对象,分析异构系统完成给定循环的能耗下界,该结果既可用于评测并行循环调度方法效率,也可用于指导异构系统体系结构优化.

本节假设处理器频率无上界约束,并放松公式(8)中迭代次数为整数的约束,因此得到原问题的一个松弛问题.如果松弛问题的解为整数,则为原问题的最优解;如果不是整数,则松弛问题的目标值为原问题目标值的下界.定理 3 给出并行循环调度的能耗下界.

定理 3. 对于有时间约束的并行循环调度问题,系统总能耗 E^*存在下界,即 $E^*\geqslant\left(\sum\limits_{1\leqslant j\leqslant m}\mu_j^{\frac{\alpha}{\alpha-1}}\right)^{1-\alpha}\dfrac{N^{\alpha}}{T^{\alpha-1}}$.

证明:$N_j(1\leqslant j\leqslant m)$表示迭代子集 I_j 的大小,因此应为区间[0,N]内的整数变量.考虑将原问题放松约束,假设 N_j 为[0,N]区间内连续变量.基于定理 2 的结论,当α=3 时,总能耗 E 可以看作是关于 $N_1,N_2,\ldots,N_m$ 的 m 元凸函数.由凸函数的性质可知,该问题最优解唯一.由于函数的最优解或出现在定义域边界或在定义域内部,下面分情况讨论:

1. 如果最优解出现在定义域边界,由循环调度约束 $\sum\limits_{1\leqslant j\leqslant m}N_j=N$ 可知,必$\exists k\in[1,m]$,有 $N_k=N$ 且 $N_i=0$ $(1\leqslant i\leqslant m,i\neq k)$.则该分配方式下总能耗 $E_{bou}=\dfrac{N^{\alpha}}{\mu_k^{\alpha}T^{\alpha-1}}$.
2. 如果最优解出现在定义域内部,可以通过 Lagrange 乘子法求解.由定理 2 可知,总能耗 E 关于处理器迭代数的函数关系为 $E=\left(\sum\limits_{1\leqslant j\leqslant m}\dfrac{N_j^{\alpha}}{\mu_j^{\alpha}}\right)\Big/T^{\alpha-1}$.循环调度约束 F 为 $\sum\limits_{1\leqslant j\leqslant m}N_j-N=0$.

与定理 1 的证明过程相似,由 Lagrange 乘子法可得 $\dfrac{\partial E}{\partial N_j}=\lambda\dfrac{\partial F}{\partial N_j}$ $(1\leqslant j\leqslant m)$.

可得 $\dfrac{\alpha}{\mu_j^{\alpha}}N_j^{\alpha-1}=\lambda$, 即 $N_j=\left(\dfrac{\lambda\mu_j^{\alpha}}{\alpha}\right)^{\frac{1}{\alpha-1}}$.

带入约束 F 表达式,可得 $\sum\limits_{1\leqslant j\leqslant m}N_j=\sum\limits_{1\leqslant j\leqslant m}\left(\dfrac{\lambda\mu_j^{\alpha}}{\alpha}\right)^{\frac{1}{\alpha-1}}=N$, 求得 $\lambda=\dfrac{\alpha N^{\alpha-1}}{\left(\sum\limits_{1\leqslant j\leqslant m}\mu_j^{\frac{\alpha}{\alpha-1}}\right)^{\alpha-1}}$.

进而求得 N_j 的表达式:

$$N_j=\frac{\mu_j^{\frac{\alpha}{\alpha-1}}}{\sum\limits_{1\leqslant k\leqslant m}\mu_k^{\frac{\alpha}{\alpha-1}}}N.$$

故有

$$E_{inn}=\sum_{1\leqslant j\leqslant m}\frac{N_j^{\alpha}}{\mu_j^{\alpha}}\Big/T^{\alpha-1}=\left(\sum_{1\leqslant j\leqslant m}\mu_j^{\frac{\alpha}{\alpha-1}}\right)^{1-\alpha}\frac{N^{\alpha}}{T^{\alpha-1}} \tag{9}$$

对于$\forall k\in[1,m]$,均满足$\mu_k^{\frac{\alpha}{\alpha-1}}<\sum_{1\leqslant j\leqslant m}\mu_j^{\frac{\alpha}{\alpha-1}}$,因此$E_{bou}>E_{inn}$.

故最优解在定义域内部,公式(9)给出的是全局最优解.由于假定 N_j 为连续变量,放松了原问题的约束条件,因此,原问题最优解 E^*应满足 $E^*\geqslant\min(E_{inn},E_{bou})=\left(\sum_{1\leqslant j\leqslant m}\mu_j^{\frac{\alpha}{\alpha-1}}\right)^{1-\alpha}\frac{N^{\alpha}}{T^{\alpha-1}}$. □

2.3 并行循环调度问题优化

高性能计算机系统往往只包含几种类型的处理器[1],但是每种类型处理器的数量很大,如果直接通过公式(8)求解最优调度问题,则需要较大的计算量.第 2.3.1 节给出合并同构处理器调度的方法以降低该问题的计算复杂度.第 2.3.2 节针对处理器频率离散可调的现状,给出了基于处理器内不同运行级的循环再调度方法,以进一步降低功耗.

2.3.1 合并同构处理器调度

公式(8)属于典型的整数规划问题,当处理器数量较大时,计算复杂度会随之增加.而减少自变量个数是降低计算复杂度的有效方法.文献[4]的研究证明,在由 m 个同构处理器组成的并行系统中,能耗最优的并行循环调度方法满足:$\forall i,j\in[1,m],|N_i-N_j|\leqslant 1$.基于该结论,我们可以合并同构处理器集合的循环调度问题.定理 4 讨论了该问题的优化求解方法.记同构处理器构成的集合为 p_{syn},其大小为 m_{syn},集合 p_{syn} 的执行速度和系统参数μ分别为 S_{syn} 和μ_{syn}.

定理 4. 如果并行系统中存在同构处理器,则可以将同构处理器集合看作整体参与调度.集合 p_{syn} 的系统参数 $\mu_{syn}=m_{syn}^{1-1/\alpha}\mu_j$,执行速度 $S_{syn}=m_{syn}S_j(j\in p_{syn})$.记简化后的调度问题中对集合 p_{syn} 分配的迭代数为 N_{syn}^*,则对集合 p_{syn} 中第 j 个处理器的能耗最优循环调度是 $N_j^*=\begin{cases}\lceil N_{syn}^*/m_{syn}\rceil, & 1\leqslant j\leqslant\omega\\ \lfloor N_{syn}^*/m_{syn}\rfloor, & \omega<j\leqslant m_{syn}\end{cases}$, $\omega=N_{syn}\bmod m_{syn}$.

证明:在由 m 个处理器组成的并行系统中,如果存在同构处理器 i 和 j,则最优循环调度必满足$|N_i-N_j|\leqslant 1$(详细证明见文献[4]).因此,我们可以将同构处理器集合的循环调度问题合并,即将同构处理器集合看作一个整体参与调度.集合 p_{syn} 的执行速度为所有处理器执行速度的累加和,即 $S_{syn}=m_{syn}S_j(j\in p_{syn})$.根据系统参数$\mu$的定义,有 $\mu_{syn}=((m_{syn}X)K^{1-1/\alpha})\big/(m_{syn}AC)^{1/\alpha}=m_{syn}^{1-1/\alpha}\mu_j$.通过将集合 p_{syn} 看作一个整体参与其余处理器调度,我们可以将公式(8)中的自变量个数减少为$(m-m_{syn}+1)$.

记简化后的规划问题中对集合 p_{syn} 分配的迭代数为 N_{syn}^*,需要进一步将迭代分配给集合 p_{syn} 中的每个处理器.根据文献[4]中对同构处理器最优循环调度的结论,能耗最优的循环调度是

$$N_j^*=\begin{cases}\lceil N_{syn}^*/m_{syn}\rceil, & 1\leqslant j\leqslant\omega\\ \lfloor N_{syn}^*/m_{syn}\rfloor, & \omega<j\leqslant m_{syn}\end{cases},\ \omega=N_{syn}\bmod m_{syn}. \tag{□}$$

2.3.2 处理器内循环再调度

定理 1 讨论了给定循环调度下的处理器最优频率选择方法.但是,目前真实处理器只支持有限个离散的频率值,因此定理 1 确定的最优频率值不一定是硬件支持的频率.已有的方法[4]是选择大于最优频率值的最小频率,虽然该方法可以满足时间约束,但是增大了功耗.基于该问题,本节提出处理器内的循环再调度方法,以进一步降低功耗.由于处理器调频操作相互独立,本节只讨论单处理器的循环再调度问题.

记第 j 个处理器支持的运行级个数为 n_j,且级数越小性能越高.第 $k(0\leqslant k\leqslant n_j-1)$个运行级的频率记为 F_j^k,相应的处理器功耗记为 P_j^k.对于定理 1 给出的最优功耗 P_j^*,如果 $P_j^*\leqslant P_j^{n_j-1}$,则全部迭代只能在第 n_j-1 个运行级完

成.如果$\exists k(0\leqslant k\leqslant n_j-2)$满足$P_j^{k+1}<P_j^*<P_j^k$,则将循环划分在$k$和$k+1$两个运行级下完成.虽然理论上可以将循环划分在更多的运行级,但是选择更多的运行级不仅会增加调压操作本身产生的额外能量开销,而且随着运行级数的增加能耗收益也会随之减小.

在使用循环再调度方法时,需要考虑调压操作的时间和能耗开销.文献[12]证明,电压调节的时间开销为$t_{switch}(V_k,V_{k+1})=\frac{2C}{I_{\mathrm{MAX}}}|V_k-V_{k+1}|$,能耗开销为$E_{switch}(V_k,V_{k+1})=(1-\upsilon)C|V_k^2-V_{k+1}^2|$.其中,$C$为切换电容,$I_{\mathrm{MAX}}$为最大电流,$\upsilon$为电压调整器的有效性因子,$V_k$和$V_{k+1}$分别是第$k$级和第$k+1$级的核心电压.

设分配在第k个运行级的迭代数为$L_k(L_k\in Z^+)$,由时间约束需满足如下不等式:

$$\frac{L_k}{X_jF_j^k}+\frac{N_j^*-L_k}{X_jF_j^{k+1}}+t_{switch}(V_k,V_{k+1})\leqslant T.$$

求解可得$L_k=\left\lceil\frac{F_j^k(N_j^*-(T-t_{switch}(V_k,V_{k+1}))X_jF_j^{k+1})}{F_j^k-F_j^{k+1}}\right\rceil$,则剩余的$N_j^*-L_k$个迭代分配在第$k+1$个运行级完成.

如果将N_j^*个迭代完全分配第k个运行级,则能耗为$E_j^k=\frac{N_j^*}{X_jF_j^k}P_j^k$;而通过循环再调度方法能耗为$E_j^{k,k+1}=\frac{L_k}{X_jF_j^k}P_j^k+\frac{N_j^*-L_k}{X_jF_j^{k+1}}P_j^{k+1}+E_{switch}(V_k,V_{k+1})$.由于处理器调压本身存在能量开销,因此该优化方法只适用于$E_j^{k,k+1}<E_j^k$的情况,化简后即为不等式$\frac{N_j^*-L_k}{\mu_j}((P_j^k)^{1-1/\alpha}-(P_j^{k+1})^{1-1/\alpha})>E_{switch}(V_k,V_{k+1})$.

3 实验与评测

本节首先介绍实验使用的软硬件平台,然后是实验中的测试用例,最后对实验结果进行了详细分析.

我们以CPU和GPU构成的异构系统为实验平台,基于该平台,我们实现了一个异构系统编译器以及一些典型应用程序[14].

3.1 编译实现

为了统一通用处理器和GPU加速器的编程模型,简化已有应用程序向CPU-GPU异构系统移植,我们基于OpenMP语言扩展了4条面向GPU的指导命令,该命令用以指导编译器将原CPU上执行的OpenMP代码转换为GPU上执行的Brook+代码.图2给出了MPtoStream编译器框架:首先,通过Parser模块分析扩展后的OpenMP程序,并且生成中间语法树;然后,通过一系列语法树转换,生成面向Brook+语言的中间语法树结构;最后,调用反向输出过程,分别生成CPU执行的OpenMP代码和GPU执行的Brook+代码.基于该框架,本文扩展了面向CPU-GPU异构系统的并行循环调度模块和处理器频率调节模块,如图2中加粗字体所示.

我们通过Profile方式获得公式(8)中的输入参数.首先将并行循环L分别在CPU和GPU上执行,测试循环在处理器不同频率下的执行时间和能量消耗,基于采样数据通过数值拟合方式计算参数μ和X.采样次数越大,拟合精度越高,但是可能会影响程序性能,具体的采样次数可以根据实际需求确定.在获得所有必要的参数后,调用整数规划算法计算最佳循环调度$F_{doall}=\langle I_1,I_2,\ldots,I_m\rangle$,然后根据定理1计算处理器运行频率.最后,通过系统支持的DVFS操作指导该循环的剩余执行过程.

图3给出了编译生成的代码示例,该程序实现了矩阵赋值操作,外层循环由1024个迭代组成.我们将循环子集(0,Ngpu−1)分配给GPU,循环子集(Ngpu,1024−1)分配给CPU执行.

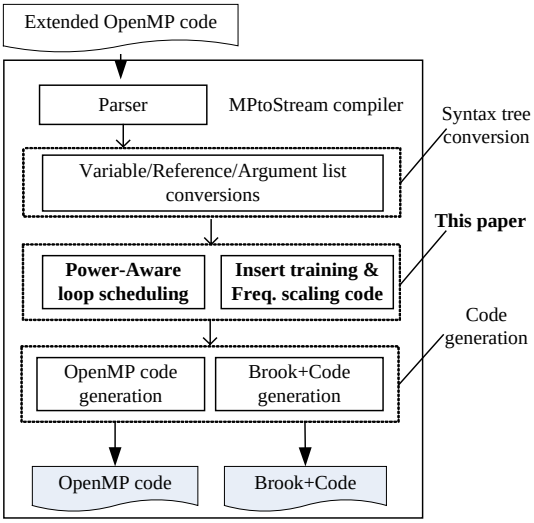


Fig.2 MPtoStream compilation framework

图 2 MPtoStream 编译框架

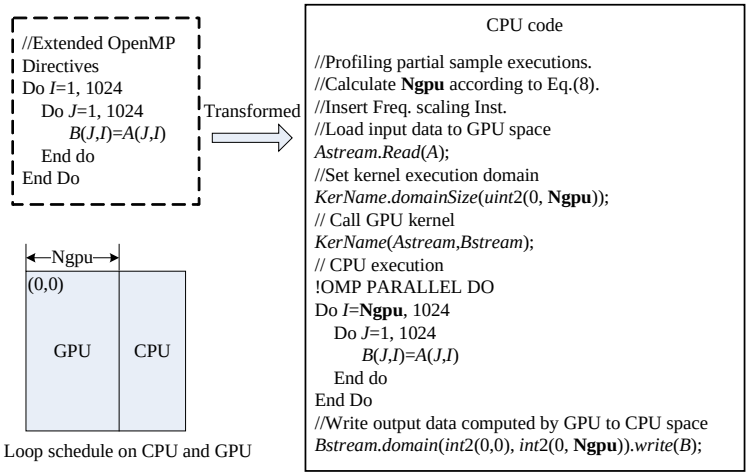


Fig.3 CPU-GPU task scheduling code after compilation

图 3 编译生成的 CPU-GPU 任务调度代码

3.2 测试平台

本文的实验平台是基于 Intel Core I7 920 Quad-Core CPU 和 AMD 4870×2 GPU 构成的异构系统.为了检验算法对异构系统的效率,我们通过调节其中一个 GPU 核心的存储器频率,将 4870×2 GPU 中两个性能相同的核心配置成两个性能不同的核心,其中,性能较高的核心记为 GPU-High.详细参数见表 1.下面简要介绍实验中的相关问题.

(1) 获取处理器支持的运行电压/频率信息.当前,主流 CPU 都支持动态调频技术,如 Intel 的 SpeedStep 和 AMD 的 PowerNow!等功耗管理技术,而且在操作系统中都集成有针对特定处理器的功耗管理模块.本文的实验平台是 OpenSUSE v10.3 64 位操作系统,通过系统提供的 ACPI 接口可以获得 CPU 支持的运行频率信息,并且可以动态调节 CPU 的运行频率.随着 GPU 逐渐走向通用计算,GPU 的功耗控制方法也逐渐完善,如 AMD 提供了

ADL 库(AMD display library),通过该接口可以获取 GPU 芯片支持的运行级别,并完成 DVFS 操作.

(2) 获取运行时系统功耗信息.我们通过外接 HIOKI 3334 功耗测试仪测量系统功耗,并且通过 RS-232 串口机制读取功耗值,该仪器的测量误差在实测值的 1dgt 范围内.本文假定处理器在不同频率下的静态功耗不变,以系统运行功耗与待机功耗的差值作为处理器运行时的动态功耗.当前,移动版处理器大都集成有侦测功耗的传感器,随着处理器功能的不断完善,通用 CPU 和 GPU 在未来都有可能提供侦测芯片功耗的传感器.这样不仅简化了功耗采集过程,而且提高了采样精度.

Table 1 Testbed description

表 1 测试平台介绍

Processor	Core I7 920 CPU	4870 GPU-high	4870 GPU-low
Core frequency (GHz)	1.6, 2.0, 2.4, 2.67	0.75, 0.65, 0.55	0.75, 0.65, 0.55
Memory frequency (GHz)	1.33 (DDR3)	0.9 (GDDR5)	0.6 (GDDR5)
Voltage (volt)	1.064~1.248	1.1~1.3	
Cache	L1 I32KB, D32KB, L2 256KB, L3 8MB	—	
Memory space	8 GB	1GB	
Compiler	Intel ifort/icc v11.1-openmp -fast	Brook+Brcc v1.4	
Operating system	OpenSUSE v10.3 64Bit, ACPI enabled		

3.3 测试用例

我们选取 10 个典型的科学计算应用程序的核心过程,见表 2.其中,314.Mgrid 和 312.Swim 均取自 SPEC OMP2001 基准程序集,Mgrid 程序实现三维 Poisson 方程求解器,测试用例中包含 Mgrid 应用的 4 个核心计算过程.Swim 程序实现二维浅水波方程求解器,我们测试该应用的 3 个核心计算过程.FDTD[13]程序完成三维 Maxwell 方程求解过程.除 FDTD 采用单精度版本外,其他程序均采用双精度版本.

Table 2 Experimental kernel description

表 2 测试用例介绍

Benchmarks	Prob. size	Origin	Benchmarks	Prob. size	Origin
RESID	256×256×256	314.Mgrid	CALC1	2048×2048	312.Swim
PSINV			CALC2		
RPRJ3			CALC3		
INTERP			FDTD	2048×128×128	Ref.[13]
GEMM	1024×1024	BLAS	LAPLACE	4096×4096	NCSA

3.4 结果分析

图 4 给出所有 Kernel 程序分别在 GPU-High 和 GPU-Low 上的执行时间和能量开销,所有结果都是以 CPU 为基准的归一化值.可以看出,大部分程序在 GPU 上都可以获得较高的加速比,只有 RPRJ3 程序在 GPU 上未能获得加速.主要原因是 RPRJ3 程序计算访存比较低,且存在跨步访问方式,而 GPU 存储器对这类非连续访问方式效率较低.可以发现,虽然 GPU 显著提升了程序性能,但是在能耗的节省量相对较小.以 INTERP 程序为例,虽然 GPU 的加速比大于 2,但是其能耗约是 CPU 的 80%;又如,RPRJ3 程序的能量开销约是 CPU 的 1.7 倍.可见,CPU 与 GPU 在不同的程序中表现出了不同的效能,因此应该根据程序的实际效能调度异构计算资源.

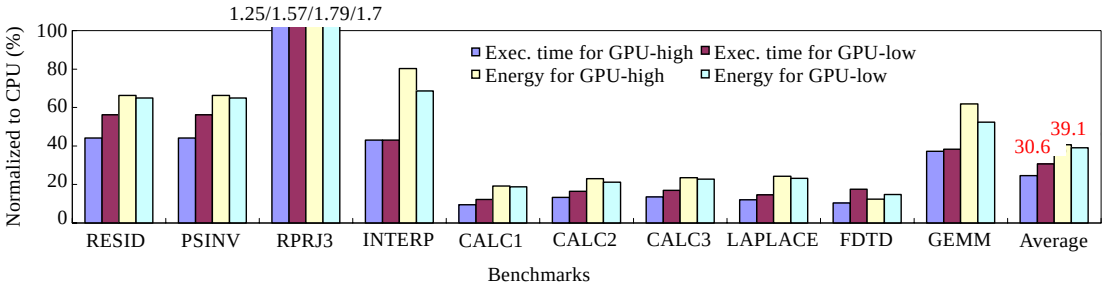


Fig.4 Comparison of execution time and energy consumption between CPU and GPU

图 4 CPU 与 GPU 执行时间和能耗开销对比

我们以单独使用峰值计算性能最高的 GPU-High 为参考,以该处理器的执行时间为约束,将 3 个处理器并行执行下的能耗和能耗延迟积(energy-delay product)与 GPU-High 的相应结果进行对比,实验结果如图 5 所示.并行执行下能耗的几何平均值约是单处理器下的 84%,能耗减少比例不大.但是通过分析我们发现,制约能耗优化效果的原因主要有:(1) 当处理器运行频率较低时,动态功耗随之降低,因而静态功耗在处理器总功耗的比例逐渐增大,限制功耗优化的整体效果;(2) 测试中使用的处理器频率范围较小,即使 3 个处理器都运行在最低频率下,也可以在约束时间前完成计算,因此能耗延迟积的优化效果较之能耗优化效果更为明显.当然,这也体现并行循环调度方法的效果.

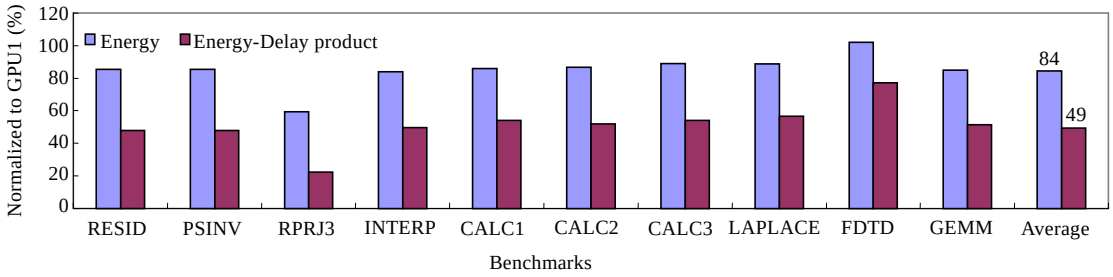


Fig.5 Energy comparison between multi-processors and single processor GPU-High

图 5 多处理器并行调度与单处理器 GPU-High 的能耗对比图

我们以 3 个处理器构成的异构系统为平台,针对每个测试程序尝试在异构处理器间进行不同的循环划分方式,测得程序的最短执行时间,记为 T.通过放松时间约束,测试执行时间不超过 $T(1+\beta)$(β称为放松因子)下的能耗优化比例.图 6 给出了当β=0%,10%,20%,30%时的能耗变化图.其中,β=0%即为无 DVFS 优化.可以看出,本文的优化算法可以有效地降低能耗,当β=10%时,所有测试程序能耗的几何平均值为优化前的 87%.同时我们也可以看出,随着时间约束的放松,总能耗减少的幅度在逐渐缩小.导致这一现象的主要原因是,随着处理器频率的逐渐降低,静态功耗在总功耗的比例逐渐增大,制约了频率降低对能耗优化的作用.通过对比不同程序的优化收益可以看出.对于访存密集型应用.如 CALC1.CALC2.CALC3.FDTD 等,其能耗减少量明显高于计算密集型应用,如 INTERP,GEMM.主要原因是.访存密集型应用的主要瓶颈是存储层次,因此在一定范围内降低处理器的频率对程序性能影响较小.

第 2.3.2 节针对处理器频率离散可调的现状,给出了处理器内循环再调度方法以进一步优化能耗.图 7 给出了放松因子β=10%时,该方法使用前后的能耗对比图,其中所列结果是实际能耗与能耗下界(见定理 3)的比值.从图中不难看出,循环再调度方法可以有效降低能耗,使实际能耗更加接近能耗下界.同时我们还应当注意,实际能耗与能耗下界依然存在差距,其主要原因是处理器频率上界约束造成实际分配结果偏离理论最优分配结果.可见,异构系统的组织对系统效能有重要影响.

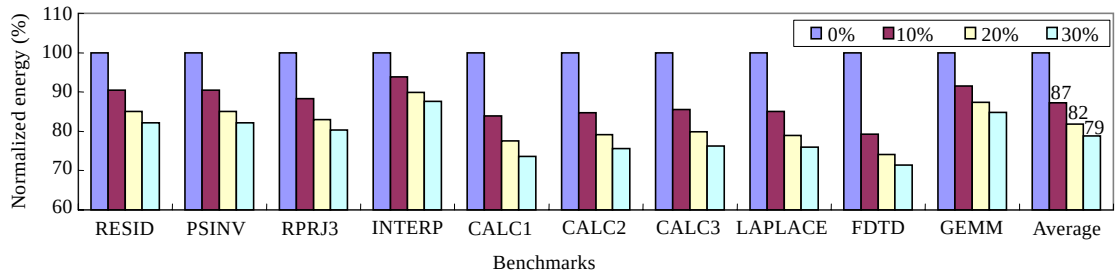


Fig.6 Energy consumption with variable performance constraint

图 6 异构系统总能耗随约束时间的变化图

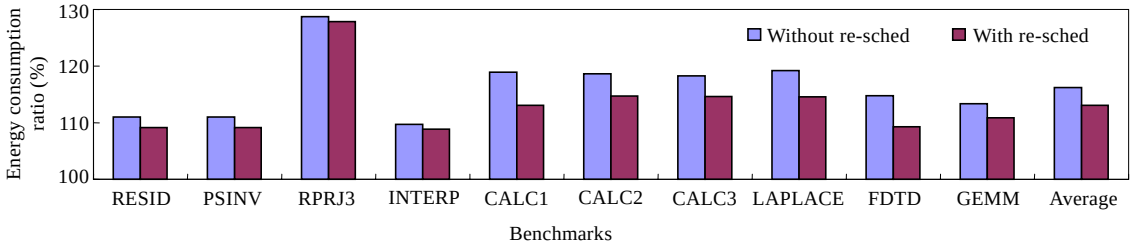


Fig.7 Energy consumption with loop re-scheduling method when β=10%

图 7 β=10%时循环再调度方法使用前后的能耗对比图

通过第 2 节的分析可知,处理器的负载和运行频率都会影响系统能耗.我们以 RESID 程序为例,观察最优并行循环调度下处理器间的负载和处理器内不同运行级下的负载随时间约束的变化情况.如图 8 所示,图中柱状图 CPU,GPU-H(GPU-high 缩写)和 GPU-L(GPU-low 缩写)中分别给出了处理器在不同运行级下的负载比例;而处理器名称后的数字表示其运行级别.可以看出,当β=10%时,3 个处理器间的负载比例与β=0%时的负载比例不同;但是当β继续增大后,处理器间的负载比例不再变化.主要原因是:公式(8)在判断处理器是否过载的依据是处理器能否在最高频率下完成任务;而对于 RESID 程序,当β=10%后,所有处理器都可以在约束时间内完成计算任务,因此处理器间的负载比例不再发生变化.与之相比,处理器内不同运行级下负载比例的变化较为明显,可见处理器频率离散问题对能耗优化的影响,再次证明了循环再调度方法的作用.

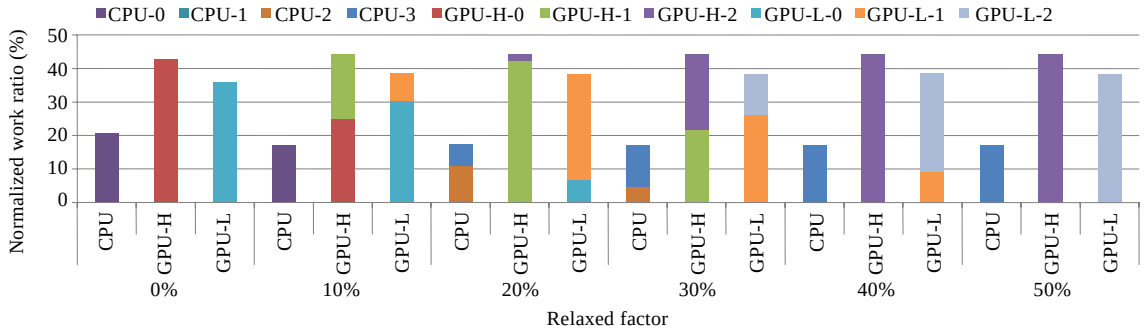


Fig.8 Work ratio allocated for each processor with variable performance constraint

图 8 处理器负载比例随约束时间的变化图

4 相关工作

本文的研究是面向异构系统的功耗优化方法,而传统同构系统的功耗优化方法研究已久.文献[3]为同构系统的功耗优化方法建立了基础理论模型,并基于该模型评估了不同任务调度方法对系统功耗的影响.但是作者的研究只针对于同构并行系统,而本文将其模型扩展到异构并行系统,并且针对类 OpenMP 并行程序,给出了异构系统能耗最优的并行循环调度方法.在技术层面,文献[5]提出关闭空闲处理器核的方法以降低执行嵌套循环的能量消耗.在作者的进一步研究[6]中,提出通过动态电压调节方法来降低轻负载处理器核的电压,以此来改善程序的负载不平衡程度,同时节省能量消耗.文献[7]提出在多处理器结构上应结合处理器数量调节和处理器电压调节两种方式来降低系统功耗.

针对处理器特征的任务调度方法,是发挥异构系统效能优势的有效途径.文献[8]根据程序在 CPU 和 GPU 上的执行时间和功耗水平,动态选择能耗较低的执行单元,优化系统的整体效能;但是,文中假设处理器的功耗与程序无关,完全由硬件决定,因此难以针对具体程序的特点进行更为有效的功耗优化,限制了算法的应用领域和实际效果.文献[9]提出一种自适应的任务调度策略,通过采样应用程序在 CPU 和 GPU 上的执行时间,分别建立执行时间关于问题规模的回归方程,然后根据执行速度选择不同执行单元的任务比例.作者研究的是针对异构系统的性能最优问题,而本文研究的是时间约束下的能耗最优问题.

随着GPU进入通用计算领域,乃至高性能计算领域,面向GPU的功耗优化技术逐渐被各大厂商和相关研究者关注.在产业领域,NVIDIA 公司提出了 PowerMizer 功耗管理方法,PowerMizer 可以根据显卡的负载情况动态调整运行频率,但是只支持相对简单的管理策略,难以根据应用的需求进行有针对性的调频操作.AMD 公司的类似功耗管理方法称为 PowerPlay.在学术领域,一些面向 GPU 的软件功耗优化方法也逐渐出现.文献[10]以一个典型的生物计算应用为案例,详细对比了 CPU 和 GPU 的执行性能和能量消耗;同时指出,GPU 高效能的发挥与程序特征和优化策略有密切关系.文献[11]评测了 GPU 在一些典型应用中的功耗开销,详细分析多处理器和存储层次等关键部分对功耗的影响.但是,在已有的GPU功耗优化方法中,尚未出现基于DVFS的编译级功耗优化方法,本文首次将该方法应用于 GPU 加速器.

5 结 论

随着 CELL,GPU 等加速器逐渐进入高性能计算领域,面向异构系统的功耗研究必将成为该领域的研究热点之一.异构系统集成有多种不同计算能力和效能关系的计算资源,为降低系统功耗提供更大的优化空间.与此同时,这也导致面向异构系统的功耗优化方法有别于传统同构系统中的优化方法.因此,本文针对类 OpenMP 并行程序,提出了异构系统功耗感知的并行循环调度方法.

本文基于 CPU-GPU 异构系统,详细评测了功耗感知的并行循环调度方法对 10 个典型应用程序的优化效果.结果表明,该方法可以有效降低系统能耗,更高效地开发了异构系统的效能优势.在未来的工作中,我们将进一步研究能耗约束情况下的并行循环调度方法,以及面向 GPU 等特定加速器的功耗优化方法.

References:

[1] http://www.top500.org

[2] http://www.green500.org/home.php

[3] Li KQ. Performance analysis of power-aware task scheduling algorithms on multiprocessor computers with dynamic voltage and speed. IEEE Trans. on Parallel and Distributed System, 2008,19(11):1484−1497. [doi: 10.1109/TPDS.2008.122]

[4] Dong Y, Chen J, Yang XJ, Deng L, Zhang XM. Energy-Oriented openmp parallel loop scheduling. In: Proc. of the 2008 IEEE Int'l Symp. on Parallel and Distributed Processing with Applications (ISPA 2008). Washington: IEEE Computer Society, 2008. 162−169. [doi: 10.1109/ISPA.2008.68]

[5] Kadayif I, Kandemir M, Karakoy M. An energy saving strategy based on adaptive loop parallelization. In: Proc. of the 39th Annual Design Automation Conf. (DAC 2002). New York: ACM, 2002. 195−200. [doi: 10.1145/513918.513968]

[6] Kadayif I, Kandemir M, Kolcu I. Exploiting processor workload heterogeneity for reducing energy consumption in chip multiprocessors. In: Proc. of the Design, Automation and Test in Europe Conf. and Exhibition (DATE 2004), Vol.2. 2004. 1158−1163. [doi: 10.1109/DATE.2004.1269048]

[7] Li J, Martinez JF. Dynamic power-performance adaptation of parallel computation on chip multiprocessors. In: Proc. of the 12th Int'l Symp. on High-Performance Computer Architecture (HPCA-12). 2006. [doi: 10.1109/HPCA.2006.1598114]

[8] Takizawa H, Sato K, Kobayashi H. SPRAT: Runtime processor selection for energy-aware computing. In: Proc. of the 3rd Int'l Workshop on Automatic Performance Tuning. 2008. [doi: 10.1109/CLUSTR.2008.4663799]

[9] Luk CK, Hong S, Kim H. Qilin: Exploiting parallelism on heterogeneous multiprocessors with adaptive mapping. In: Proc. of the 42nd Annual IEEE/ACM Int'l Symp. on Microarchitecture (MICRO 42). New York, 2009.

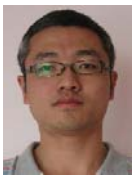
[10] Huang S, Xiao S, Feng W. On the energy efficiency of graphics processing units for scientific computing. In: Proc. of the 2009 IEEE Int'l Symp. on Parallel&Distributed Processing (IPDPS 2009). 2009. [doi: 10.1109/IPDPS.2009.5160980]

[11] Collange S, Defour D, Tisserand A. Power consumption of GPUs from a software perspective. In: Allen G, Nabrzyski J, Seidel E, van Albada GD, Dongarra J, Sloot PMA, eds. Proc. of the 9th Int'l Conf. on Computational Science: Part I. LNCS 5544, Baton Rouge, Berlin, Heidelberg: Springer-Verlag, 2009. 914−923. [doi: 10.1007/978-3-642-01970-8_92]

[12] Burd TD, Brodersen RW. Design issues for dynamic voltage scaling. In: Proc. of the 2000 Int'l Symp. on Low Power Electronics and Design, ISLPED 2000. New York: ACM, 2000. 9−14. [doi: 10.1109/LPE.2000.155245]

[13] http://developer.amd.com/SAMPLES/STREAMSHOWCASE/Pages/default.aspx

[14] Wang GB, Tang T, Fang XD, Ren XG. Program optimization of array-intensive SPEC2k benchmarks on multithreaded GPU using CUDA and Brook+. In: Proc. of the 15th Int'l Conf. on Parallel and Distributed Systems (ICPADS 2009). 2009. 292−299. [doi: 10.1109/ICPADS.2009.12]



王桂彬(1981－),男,湖南长沙人,博士生,主要研究领域为低功耗编译优化技术.

林一松(1983－),男,博士生,主要研究领域为 GPU 性能与功耗优化.

杨学军(1963－),男,博士,教授,博士生导师,CCF 会员,主要研究领域为并行体系结构,低功耗编译,流计算,容错计算.

李鑫(1984－),男,博士生,主要研究领域为计算机体系结构,云计算.

徐新海(1983－),男,博士生,主要研究领域为计算机体系结构,容错技术.