软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(10):2460−2472 [doi: 10.3724/SP.J.1001.2013.04357]

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

一种面向异构并行系统的最大功耗管理方法*

王桂彬[1], 杜 静[2], 唐 滔[1]

[1](国防科学技术大学 计算机学院,湖南 长沙 410073)

[2](电子信息系统复杂电磁环境效应国家重点实验室,河南 洛阳 471000)

通讯作者: 王桂彬, E-mail: wgbljl@gmail.com

摘 要: 高功耗已成为制约高性能计算机发展的重要问题之一.近年来,大量研究关注于如何在满足系统功耗约束的条件下优化系统执行性能.然而,已有方法大都针对同构系统,未考虑异构处理器之间的功耗或速度差异,难以高效应用于基于加速器的异构系统.对当前异构并行系统执行模型进行了抽象,并提出了融合两级功耗控制机制的系统功耗管理框架,自顶向下依次为系统级功耗控制器和异构处理引擎功耗控制器.在异构处理引擎功耗控制中,针对类 OpenMP 并行循环,首先分析了异构多处理器在满足功耗约束条件下达到性能最优的条件.基于该结果,给出了功耗受限的并行循环划分算法,该方法通过协调并行循环调度和动态电压频率调节技术以优化异构并行处理.在系统级功耗控制中,建立了异构处理引擎效能评估方法,以此作为功耗划分的依据,在兼顾并发应用公平性的同时,提高系统整体执行效能.最后,基于典型 CPU-GPU 异构系统验证了方法的有效性.

关键词: 异构并行系统;最大功耗管理;任务调度;动态电压/频率调节

中图法分类号: TP302 **文献标识码**: A

中文引用格式: 王桂彬,杜静,唐滔.一种面向异构并行系统的最大功耗管理方法.软件学报,2013,24(10):2460−2472. http://www. jos.org.cn/1000-9825/4357.htm

英文引用格式: Wang GB, Du J, Tang T. Peak power management method for heterogeneous parallel system. Ruan Jian Xue Bao/Journal of Software, 2013,24(10):2460−2472 (in Chinese). http://www.jos.org.cn/1000-9825/4357.htm

Peak Power Management Method for Heterogeneous Parallel System

WANG Gui-Bin[1], DU Jing[2], TANG Tao[1]

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

[2](State Key Laboratory of Complex Electromagnetic Environmental Effects on Electronics and Information System, Luoyang 471000, China)

Corresponding author: WANG Gui-Bin, E-mail: wgbljl@gmail.com

Abstract: High power consumption has become one of the top considerations in high performance computing field. Recently, there are many studies focused on optimizing the system performance within a given power budget. However, most existing solutions are explored for homogeneous system without considering the differences in power consumption and processing speed between heterogeneous processors, and therefore could not be adapted for accelerator-based heterogeneous parallel system effectively. This paper first summarizes the execution model of modern accelerator-based parallel system, and then introduces the power control framework consisting of two power management hierarchies, i.e. system-level power controller and heterogeneous processing engine power controller from top to bottom. In the lower level controller, targeted for OpenMP-like parallel loop, the paper first theoretically analyzes the conditions for the maximum performance given a power budget for heterogeneous processors. Based on this result, the paper provides a power-constrained parallel loop scheduling algorithm which coordinates parallel loop partition and voltage/frequency scaling for heterogeneous processers to achieve the optimal performance given a system power budget. In the upper level controller, the paper establishes the evaluation metrics

* 基金项目: 国家高技术研究发展计划(863)(2012AA01A301); 国家重点基础研究发展计划(973)(2011CB309705-1); 国家自然科学基金(60903059, 61303063)

收稿时间: 2012-09-01; 修改时间: 2012-10-19; 定稿时间: 2012-12-03

for heterogeneous processing engine to allocate power budget, in order to keep fairness between concurrent applications and improve the whole system efficiency. Finally, the paper evaluates the proposed method in a typical CPU-GPU system.

Key words: heterogeneous parallel system; peak power management; task scheduling; dynamic voltage/frequency scaling

异构并行系统通过集成通用处理器和高效能(单位功耗的计算性能)专用处理器,在具备高峰值性能的同时,有效地提高了系统整体效能,已成为高性能计算机发展的重要趋势之一.在 2012 年 6 月发布的 TOP500 超级计算机排名中,前 10 台超级计算机中有 4 台采用异构体系结构模型[1].作为首台千万亿次异构并行系统,Tianhe-1A 系统采用 CPU-GPU(graphics processing units)混合结构,系统效能达到 635.15MFLOPS/W.此外,Intel 发布的 Discovery 系统,通过集成 Intel MIC(many integrated core)加速芯片,系统效能甚至达到 1.17GFLOPS/W.

虽然异构并行系统具有更高的峰值效能,但是系统绝对功耗依然很高.过高的功耗给系统封装、供电和散热带来极大的挑战,因此,功耗不仅是系统优化的重要目标,也逐渐成为决定系统设计的重要约束条件之一[2].在 2012 年 6 月发布的 TOP500 排名中,前 5 台高性能计算机的系统平均功耗达到 6.39MW.因此,近年来国际上已有大量研究[3−7]探索如何在满足系统最大功耗约束条件下优化系统执行性能的问题,称为最大功耗管理方法(peak power management).然而,已有的功耗管理方法大都面向同构并行系统,未考虑不同计算单元间的速度或功耗差异,使之难以高效应用于基于加速部件的异构并行系统.

本文通过对当前典型异构并行系统的执行模型进行抽象,提出了融合多级功耗控制器的系统功耗管理框架,旨在满足给定系统功耗约束条件下挖掘异构并行处理的优势,优化系统执行性能.本文的主要贡献包括:

(1) 针对异构并行系统执行模型,提出了融合系统级功耗控制和异构处理引擎功耗控制的系统功耗管理框架;

(2) 针对异构处理引擎功耗控制,提出了应用感知的最大功耗管理方法.通过分析方法给出了异构处理器在给定功耗约束前提下达到性能最优的条件.基于该结果,给出了功耗受限的并行任务划分算法,该方法通过协调并行任务划分和动态电压频率调节技术优化异构并行处理效率;

(3) 针对系统级功耗控制,建立了异构处理引擎效能评估方法,以此作为功耗划分的依据,在兼顾并发应用公平性的同时,提高系统整体执行效能;

(4) 基于典型 CPU-GPU 异构并行系统,通过 4 个典型应用验证了本文方法的有效性.

本文第 1 节介绍异构系统模型以及本文提出的功耗控制整体框架.第 2 节和第 3 节分别讨论异构处理引擎和系统级功耗控制方法.第 4 节给出实验评测与分析.第 5 节介绍目前国内外功耗领域的最新研究进展.第 6 节总结本文工作,并给出今后有待研究的问题.

1 系统模型假设与功耗控制框架

本节首先对当前异构并行系统结构及其执行模式进行抽象,建立本文研究的模型基础.在此基础上,介绍本文提出的异构系统功耗控制整体框架.

图 1 给出了一个典型的异构并行系统示意图.未来异构并行系统可能包含多种计算资源,包括通用 CPU,GPU,MIC 或 FPGA(field programmable gate-array)[8].通常,加速部件只负责执行特定计算任务,而不具备完整的任务管理和调度机制,因此,加速部件大都需要在通用微处理器(主处理器)的控制下执行.如图 1 所示,该系统由 4 个计算模块组成,每个计算模块中均包含有数个主处理器和多种加速部件.

在当前主流异构并行编程模型中,多线程并行程序的每个线程应映射在主处理器上,而主处理器负责将线程内的特定计算过程加载到加速部件上执行.本文将执行同一应用程序的主处理器和加速部件构成的处理器集合称为异构处理引擎(heterogeneous processing engine,简称 HPE).以图 1 为例,系统中有两个并行程序并发执行,程序 0 占用一个计算模块,而程序 1 占用两个计算模块,系统中其他计算单元处于空闲状态.

与图 1 给出的异构并行系统模型相对应,本文提出的异构系统最大功耗管理框架采用两层控制结构,如图 2

所示.

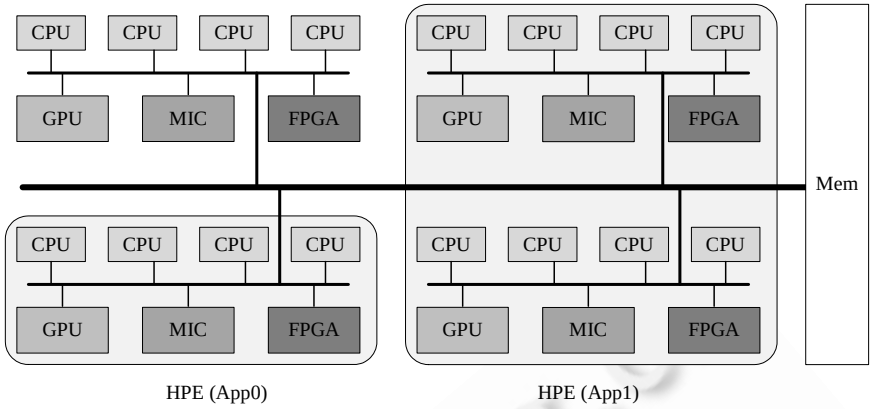


Fig.1 A typical heterogeneous parallel system

图 1 异构并行系统示意图

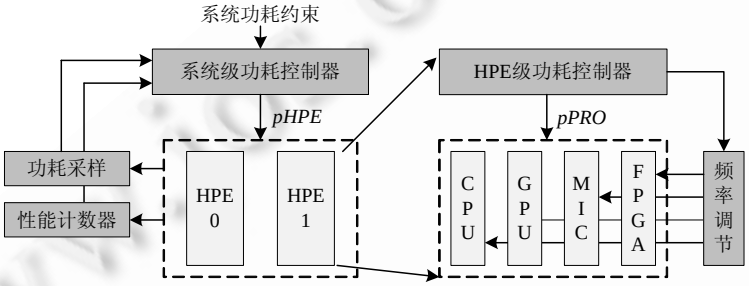


Fig.2 Power control framework for heterogeneous parallel system

图 2 异构并行系统功耗控制框架

首先,系统级功耗控制器根据给定的系统功耗约束,将功耗总额分配给并发执行的多个并行程序(HPE).其分配策略既应考虑不同应用程序对系统整体效能的贡献,同时也应兼顾并发应用的公平性,避免部分应用由于得不到足够的功耗而被饿死.系统级控制器不关心应用程序的具体执行过程,因此采用应用透明的控制方式,即以固定周期(记为 T_c)执行功耗控制过程.

其次,异构处理引擎功耗控制器负责将功耗 *pHPE* 分配给引擎内的异构处理器.由于 HPE 中异构处理器具有不同的执行速度和功耗开销,因此,如何在满足功耗约束 *pHPE* 的条件下优化 HPE 的整体性能是问题的关键.本文以类 OpenMP 并行程序为对象,结合并行循环在异构多处理器上的循环调度与频率调节技术,以发掘异构并行处理的优势.

可以看出,异构并行系统功耗控制的重点和难点在于 HPE 级功耗控制器,而已有方法[3−7]均针对同构系统,并采用应用透明的方式执行控制过程,这些均难以高效开发异构并行处理优势.因此,本文的重点是研究 HPE 级功耗控制方法.综上所述,后文第 2 节和第 3 节将分别介绍异构处理引擎和系统级功耗控制机制和实现策略.

2 异构处理引擎功耗控制方法

目前,针对系统最大功耗管理的研究[4−7]大都基于固定时间间隔的管理方法,采用对应用透明的控制策略.虽然该方法具有较好的移植性,但却忽略了程序在异构处理器上的执行特性.考虑异构处理器间的功耗和速度差异,如何协调不同处理器的负载和运行频率,以在满足功耗约束的条件下最大化并行执行速度,是 HPE 级功耗

控制的核心.针对该问题,本文提出了编译级应用感知的功耗管理方法.

如前文所述,本文的研究主要针对类 OpenMP 并行程序.OpenMP 程序主要由串行段和以编译指导语句标识的并行段组成.因此,本文以串行段和并行段的入口点作为应用感知的功耗控制器的调用点,如图 3 所示.而传统的应用透明的功耗控制方法以固定的时间间隔监测系统功耗情况,制定功耗划分策略,而未考虑任务划分策略对异构并行系统功耗优化的影响.下面详细介绍应用感知的功耗控制方法的实现策略.

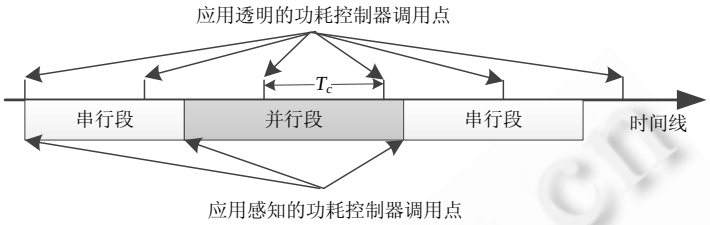


Fig.3 Timeline of invocations of application-aware power controller

图 3 应用感知的功耗控制器调用时序图

综上所述,本节以类 OpenMP 的并行程序为研究对象,研究在功耗受限的条件下,通过并行循环调度和处理器频率调节方法优化异构并行系统整体性能.首先,我们介绍处理器模型假设以及功耗优化的相关概念.

处理器功耗主要由动态功耗和静态功耗组成,其中,动态功耗与处理器运行频率有关,可通过动态电压频率调节技术降低功耗开销.处理器静态功耗与运行电压和芯片温度有关,并且在处理器运行过程中始终存在.本文的研究主要针对动态功耗部分,因此假设静态功耗在系统运行过程中保持不变.除特别说明外,后文中提到的功耗均指动态功耗.

不失一般性,假设 HPE 由 m 类处理器组成,记为 $R=\{r_0,\ldots,r_{m-1}\}$,其中,第 i 类处理器 r_i 的数量记为 N_i,其在最高频率下的动态功耗和速度分别记为 P_i 和 V_i,其中,速度是指处理器单位时间内完成的任务量.根据 CMOS 电路功耗公式,处理器动态功耗与运行电压和频率的关系可以表示为 $p=AU^2f$,其中,A 是有效切换电容,U 是核心电压,f 是运行频率.而运行频率与核心电压的关系为

$$f=K(U-U_T)^{\gamma}/U(1\leqslant\gamma\leqslant 2),$$

其中,U_T 为阈值电压,K 和 γ 是与工艺相关的参数.一般 U_T 很小,因此频率与电压的关系近似表示为 $f=KU^{\gamma-1}$.综上所述,动态功耗与频率的关系可表示为 $p=Cf^{\alpha}$,其中,C 是与体系结构相关的常数,系数 α 在 2 到 3 之间[9].由此可知,处理器动态功耗与运行频率的 α 次方成正比关系.

在异构多处理器系统中,不同处理器的最大运行频率可能各不相同,因此,我们以相对运行频率代替处理器的实际运行频率,即第 i 类处理器的实际动态功耗 $p_i=f_i^{\alpha}P_i$,其中,f_i 为处理器相对运行频率,即处理器实际运行频率与其最高频率的比值.

并行循环是不存在跨迭代依赖的循环结构,因此可以在多处理器上并行执行.设并行循环的迭代次数为 N,并假设其在 m 类处理器核心上并行执行.由于所有处理器执行相同的循环迭代过程,因此我们以单位时间内完成的迭代数描述处理器的执行速度.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:将并行循环的迭代空间加以划分,并映射到异构处理器上,记为

$$F_{doall}=\langle I_0,I_1,\ldots,I_{m-1}\rangle,$$

其中,$I_i(0\leqslant i\leqslant m-1)$ 表示并行循环迭代集合的子集,且被映射到第 i 类处理器上;$n_i(n_i\in N^+)$ 表示子集 I_i 的大小,即迭代次数.

在上述划分方式下,异构多处理器并行完成该循环的时间为

$$T=\max\left\{\left.\frac{n_i}{N_iv_i}\right|0\leqslant i\leqslant m-1\right\};$$

而各处理器的总功耗为

$$P=\sum_{i=0}^{m-1}N_i p_i,$$

其中,v_i 和 p_i 表示第 i 类处理器的实际执行速度和运行功耗,因此满足 $v_i\leqslant V_i, p_i\leqslant P_i$.

HPE 级的功耗管理问题可以描述为:在满足 $P\leqslant pHPE$ 的条件下,最小化总执行时间 T.不难发现,该问题可以归纳为典型的规划问题.

下面我们试图通过放松该问题的约束条件,探索异构并行处理达到性能最优的条件.

我们放松 $n_i(0\leqslant i\leqslant m-1)$为正整数的约束条件,即假设并行循环任务可连续划分.不难发现,所有处理器的执行时间均相等是最优解的必要条件.可知,在总任务量一定的情况下,性能最优化问题可以转化为总执行速度最大化问题,即最大化 HPE 的总执行速度 $V=\sum_{i=0}^{m-1}N_i v_i$.下面的定理分析了异构并行处理达到性能最优的条件.

定理(功耗约束下异构并行处理最优性能定理). 将并行计算任务分配给异构多处理器并行执行,根据系统最大功耗约束 $pHPE$ 的取值,当异构多处理器达到性能最优时,处理器间满足如下关系:

(1) 如果 $pHPE\leqslant\rho\psi^{-\alpha/(\alpha-1)}$,则各处理器动态功耗满足 $\frac{p_i}{p_j}=\frac{\beta_i^{-1}}{\beta_j^{-1}}$;

(2) 如果 $pHPE>\rho\psi^{-\alpha/(\alpha-1)}$,则必有部分类型的处理器运行在最高频率,且剩余类型处理器集合中各处理器满足 $\frac{p_i}{p_j}=\frac{\beta_i^{-1}}{\beta_j^{-1}}$,

其中,$\rho=\sum_{i=0}^{m-1}N_i\beta_i^{-1}$, $\beta_i=(P_i/V_i^{\alpha})^{1/(\alpha-1)}$, $\psi=\max\{V_i/P_i\mid 0\leqslant i\leqslant m-1\}$.

证明:

异构处理引擎总执行速度可以表示为

$$V=\sum_{i=0}^{m-1}N_i V_i f_i=\sum_{i=0}^{m-1}N_i V_i(p_i/P_i)^{1/\alpha}=\sum_{i=0}^{m-1}\frac{V_i N_i}{P_i^{1/\alpha}}p_i^{1/\alpha};$$

而动态功耗约束可以描述为

$$F: pHPE-\sum_{i=0}^{m-1}N_i p_i=0.$$

可知,总执行速度和功耗约束都是关于变量 p_i 的函数,可通过 Lagrange 方法求解极值.

令 $\frac{\partial V}{\partial p_i}=\kappa\frac{\partial F}{\partial p_i}$ (κ为 Lagrange 乘子),可得:

$$\frac{1}{\alpha}\frac{V_i}{P_i^{1/\alpha}}p_i^{1/\alpha-1}=\kappa,$$

可转化为

$$\frac{V_i}{P_i^{1/\alpha}}p_i^{1/\alpha-1}=\frac{V_j}{P_j^{1/\alpha}}p_j^{1/\alpha-1},$$

即

$$\frac{p_i}{p_j}=\frac{\beta_i^{-1}}{\beta_j^{-1}},$$

其中,$\beta_i=(P_i/V_i^{\alpha})^{1/(\alpha-1)}$.

由上式可知,在总功耗一定的情况下,当处理器动态功耗满足 $\frac{p_i}{p_j}=\frac{\beta_i^{-1}}{\beta_j^{-1}}$ 时,异构并行处理整体性能达到最优.此时, $p_i=\frac{\beta_i^{-1}}{\rho}pHPE$,其中, $\rho=\sum_{i=0}^{m-1}N_i\beta_i^{-1}$.

需要注意的是,以上分析成立的条件是对于任意处理器 r_i,均有 $p_i\leqslant P_i$.

此时,最大功耗约束应满足 $pHPE\leqslant\rho\psi^{-\alpha/(\alpha-1)}$,其中,$\psi=\max\{V_i/P_i\mid 0\leqslant i\leqslant m-1\}$.

下面分析当 $pHPE>\rho\psi^{-\alpha/(\alpha-1)}$时的情况.此时,对于部分处理器,公式 $p_i=\frac{\beta_i^{-1}}{\rho}pHPE$ 给出的最优功耗值必然超出其最大功耗,因此,问题的关键是这类处理器的功耗分配问题.由于速度 V 关于 p_i 的二阶导数 $\frac{\partial^2 V}{\partial^2 p_i}$ 恒小于 0,由凸优化性质可知,如果极值点 p_k 大于实际最大值 P_k,则 p_k 应等于 P_k,即处理器 r_k 应运行在其最高频率.此后,再将剩余功耗 $pHPE-N_kp_k$ 分配给其他处理器.重复上述过程,直至满足第 1 种情况的条件终止.综上所述,当 $pHPE>\rho\psi^{-\alpha/(\alpha-1)}$时,必有部分处理器运行在最高频率,而剩余处理器的功耗依然满足 $\frac{p_i}{p_j}=\frac{\beta_i^{-1}}{\beta_j^{-1}}$. □

上述定理给出了不同功耗约束下异构处理器达到性能最优的条件.基于该定理,我们推导出异构处理器达到性能最优时满足的性质.

推论. 在功耗约束的条件下,当异构处理器达到性能最优时,所有未达到最大运行频率的处理器具有效能相等的关系,即 $\frac{v_i}{p_i}=\frac{v_j}{p_j}$ $(0<f_i,f_j<1)$.

证明:对于任意两个未达到最大运行频率的处理器 r_i 和 $r_j(0<f_i,f_j<1)$.由以上定理可知,当总执行速度达到最优时,处理器满足 $\frac{p_i}{p_j}=\frac{\beta_i^{-1}}{\beta_j^{-1}}$,即:

$$\frac{p_i}{p_j}=\frac{P_i f_i^{\alpha}}{P_j f_j^{\alpha}}=\frac{\beta_i^{-1}}{\beta_j^{-1}}.$$

故有:

$$\frac{f_i}{f_j}=\left(\frac{\beta_i^{-1}P_j}{\beta_j^{-1}P_i}\right)^{1/\alpha}=\left(\frac{P_j\beta_j}{P_i\beta_i}\right)^{1/\alpha}.$$

因此,

$$\frac{v_i}{v_j}=\frac{V_i f_i}{V_j f_j}=\frac{V_i}{V_j}\left(\frac{P_j\beta_j}{P_i\beta_i}\right)^{1/\alpha}=\frac{\beta_i^{-1}}{\beta_j^{-1}}=\frac{p_i}{p_j},$$

即:

$$\frac{v_i}{p_i}=\frac{v_j}{p_j}.$$

由上式可知,在总功耗一定的条件下,当异构并行系统达到性能最优时,所有未达到最高频率的处理器的实际效能相同. □

根据以上推论可知,异构处理器最优功耗管理技术的核心思想是使各处理器达到效能相等.由效能的定义 $(v/p=f^{1-\alpha}V/P)$可知,降低运行频率可以提高执行效能,因此,随着系统约束功耗的缩小,效能较高的处理器核心将优先达到其最大频率.

上述定理分析了异构多处理器达到性能最优的条件,基于此,我们提出了功耗受限下性能最优的异构处理器并行任务划分算法 PcPar,如图 4 所示.算法的输入是异构处理器的配置信息和给定功耗约束,算法的输出是划分各处理器的循环迭代数以及处理器运行频率.

算法 PcPar 的主要流程介绍如下:首先,根据定理计算各处理器的最优运行频率 f_i^* (第 2 行~第 7 行),其主要过程是反复判断剩余功耗配额 $pHPE$ 是否满足定理中第 1 种情况的约束条件.如果不满足该条件,则将当前集合 R 中效能最高的处理器置于最高运行频率;如果此时剩余功耗小于 $\rho\psi^{-\alpha/(\alpha-1)}$,则集合 R 中处理器的运行频率应小于其最大频率,此时,可以通过公式 $p_i=\frac{\beta_i^{-1}}{\rho}pHPE$ 计算出最优功耗值(第 7 行).由于实际处理器只支持离散频率值,因此应选择不超过 f_i^* 的最大物理频率 f_i(第 8 行).如果此时各处理器的功耗总和小于功耗约束,则在满

足功耗约束的条件下来提升部分处理器的运行频率(第 9 行).为了最大化并行执行性能,算法根据不同类型处理器的实际运行速度,按比例分配并行循环迭代,即令 $n_i \propto N_i V_i f_i$(第 10 行).

Algorithm PcPar. Power-Constrained Parallel Loop Schedule.
Input: Peak Power *pHPE*, Total Iterations *N*, Processor List $R=\{r_i=\langle P_i,V_i,N_i\rangle\}$;
Output: Iterations $\{n_i\}$ and operating frequency $\{f_i\}$ for each processor set.

1. Set $\psi=\max\{V_i/P_i|r_i\in R\}$ and $\rho=\sum_{r_i\in R} N_i\beta_i^{-1}$, where $\beta_i=(P_i/V_i^{\alpha})^{1/(\alpha-1)}$.
2. while $pHPE>\rho\psi^{-\alpha/(\alpha-1)}$
3. Find processor k achieving $V_k/P_k=\max\{V_i/P_i|r_i\in R\}$.
4. Set $f_k^*=1$, $pHPE=pHPE-N_kP_k$.
5. Set $R=R-\{r_k\}$, re-compute ρ and ψ.
6. end
7. Set $p_i^*=\frac{\beta_i^{-1}}{\rho}pHPE$, $f_i^*=(p_i^*/P_i)^{1/\alpha}$, for each $r_i\in R$.
8. Choose the maximum hardware-supported frequency f_i not exceeding f_i^* $(0\leqslant i\leqslant nPRO-1)$.
9. Promote the operating frequency f_i, if there exists underutilized power budget.
10. Set the iterations $n_i=\left\lfloor\frac{N_iV_if_i}{\sum_{k=0}^{nPRO-1}N_kV_kf_k}\times N\right\rfloor$ $(0\leqslant i<nPRO-1)$ and $n_{nRPO-1}=N-\sum_{i=0}^{nPRO-2}n_i$

Fig.4 Power-Constrained parallel loop scheduling algorithm

图 4 功耗受限的并行循环调度算法

3 系统功耗控制方法

系统级功耗控制方法的目的是在满足系统功耗约束的条件下尽可能地提高系统整体的执行效率,同时应尽可能地保证并发应用间的公平性.本文采用效能优先的系统级功耗分配策略,下面首先介绍程序效能的评估方法,我们首先给出 HPE 的效能评估方法.

并发执行的多道程序具有不同的程序特征和执行轨迹,因此我们以每秒完成的指令数(billion instructions per second,简称 BIPS)描述 HPE 的执行速度.需要注意的是,由于不同类型的处理器具有不同的指令集体系结构(instruction set architecture),因此,直接比较不同类型处理器的 BIPS 值有失公平.为此,我们首先将加速部件(GPU 等)的运行速度归一化为主处理器(CPU)的速度,在同一基准衡量 HPE 的执行性能.记 HPE 中 CPU 每秒完成的指令数为 BIP_0,该值可通过硬件计数器获得[10].根据第 2 节的分析,可以获得主处理器和任意加速部件在当前运行频率下的执行速度,分别记为 v_0 和 $v_k(1\leqslant k\leqslant nPRO-1)$,可得加速部件等效每秒完成的指令数 $BIP_k=BIP_0\times v_k/v_0$.可知,单个 HPE 的速度为

$$bHPE=BIP_0\times\sum_{k=0}^{nPRO-1}v_k/v_0,$$

其中,*nPRO* 表示异构处理引擎中包含的处理器数量.

由于应用本身所具有的特征,不同的应用程序具有各不相同的 BIPS,因此直接根据绝对 BIPS 值划分系统功耗,会恶性地延长部分本身具有较小 BIPS 的应用程序的执行时间,难以保证并发任务间的公平性.与文献[7]中的评估方法相同,我们以相对执行速度描述程序执行特征,相对执行速度定义为当前频率下 BIPS 与最高频率下 BIPS 的比值,即:

$$rbHPE=bHPE/bHPE^*,$$

其中,$bHPE^*$表示所有处理器均运行在最高频率下 HPE 的 BIPS 值.记该频率下 HPE 的功耗为 *pHPE*,则效能定义为相对执行速度与功耗开销的比值,即:

$$eHPE=rbHPE/pHPE.$$

在获得各 HPE 的效能值的基础上,我们将总功耗 P_{sys} 按照各程序的效能比分配给多个 HPE,即分配给第 i 个 HPE 的总功耗为

$$pHPE_i = \frac{eHPE_i}{\sum_{k=0}^{nHPE-1} eHPE_k} P_{sys},$$

其中,$nHPE$ 为系统中并发执行的程序数量.

根据上述功耗分配策略,系统级功耗控制器以固定的周期 T_c 执行功耗控制过程,即根据各应用程序在前一个控制周期的运行效能 $eHPE_i$,结合当前系统功耗约束 P_{sys},计算各应用程序在下一个控制周期的功耗约束,并传递给下一级功耗控制器.

4 实验评估

本文以 Intel Core I7 920 Quad-Core CPU 和 AMD 4870×2 GPU 构成的异构系统为实验平台,具体的系统参数配置参见表 1,这里不再赘述.

Table 1 Experimental platform

表 1 测试平台

Processor	Intel Core I7 920 CPU	AMD 4870×2 GPU
Core frequency (GHz)	2.67, 2.4, 2.0, 1.6 (4 levels)	0.75, 0.65, 0.55 (3 levels)
Memory frequency (GHz)	1.33 (DDR3)	0.9 (GDDR5)
Cache	L1 I32KB, D32KB, L2 256KB, L3 8MB	–
Memory space	8 GB	1GB
Compiler	GCC v4.2.1 -fopenmp -O3	AMD Stream SDK1.4, APP SDK 2.2
Operating system	OpenSUSE v10.3 ×86_64, ACPI enabled	

实验使用的主处理器由 4 个处理器核构成,因此我们令 1 个处理器核心负责调度和管理 GPU,而令剩余 3 个处理器核心处理 OpenMP 并行循环,该方法与文献[11]中提出的异构并行系统执行方式相同.在实验中需要获取处理器执行速度,即每秒完成的指令数 $BIPS$,该值可通过 Intel 公司提供的性能计数器(performance counter monitor,简称 PCM[10])获得.本文假定系统的控制周期 T_c 为 1s,因此我们设置 PCM 以秒为单位监测处理器的执行情况.

实验中,通过外接 HIOKI 3334 功耗测试仪测量系统功耗,并通过 RS-232 串口机制读取功耗值,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.我们首先通过单独测试以获得不同类型的处理器在运行特定程序时的功耗,然后以该结果作为后续优化算法的输入;通过测量优化后的程序运行结果,检验算法的有效性.

4.1 测试用例

我们选取 4 个计算密集型应用作为测试用例,见表 2.MGRID 和 SWIM 均选自 SPECOMP2001 基准测试集,分别实现了泊松方程求解器和浅水波模型求解器.原程序通过 OpenMP 并行编程语言实现,通过 Brook+语言将其映射在 GPU 上执行.HotSpot 和 Kmeans 均选自 Rodinia 程序集.Rodinia[12]程序集主要用于异构并行系统评估,因此提供了 OpenMP,OpenCL 等多种实现版本,其中,HotSpot 应用实现芯片温度模型的模拟,*K*-means 实现数据挖掘领域经常使用的聚合算法.

Table 2 Experimental applications

表 2 测试用例

程序名称	选自	描述	问题规模	核心程序
HotSpot	Rodinia	Thermal simulation tool	2048×2048 data points	HotSpot
K-Means		Clustering algorithm	819 200 points 34 features	Cluster
MGRID	SPECOMP2001	Poisson equation solver	256×256×256 data points	RESID, PSINV, RPRJ3, INTERP
SWIM		Shallow water modeling solver	2048×2048 data points	CALC1, CALC2, CALC3

4.2 实验结果与分析

在检验本文提出的最大功耗管理方法之前,我们首先评估了各 Kernel 程序在 CPU 和 GPU 上的功耗开销和执行性能.

图 5 给出了各 Kernel 程序在 CPU 和 GPU 上的功耗开销.与通用微处理器相比,GPU 加速部件具有较高的功耗开销,其运行功耗普遍超过 100W(99W~128W),成为系统中功耗开销最大的组成部分之一.随着 GPU 加速部件在高性能计算领域应用的不断普及,其功耗控制与优化问题将得到更多的关注.对比不同 Kernel 程序的执行情况可以发现,CPU 的功耗开销变化区间较小,而 GPU 在不同的 Kernel 程序中具有较大的功耗变化区间.

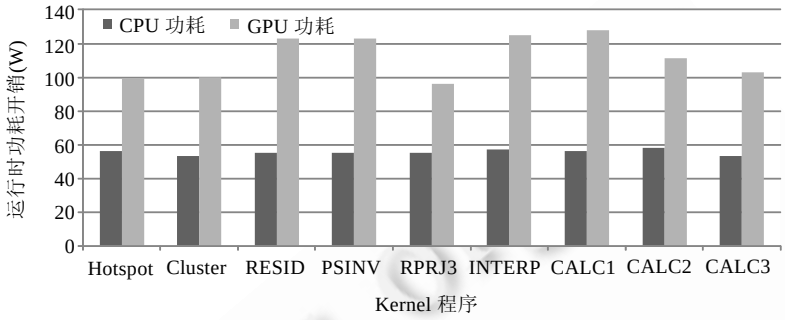


Fig.5 Power consumption comparison for heterogeneous processors

图 5 异构处理器功耗开销对比图

图 6 给出了各程序在 CPU 上的实际性能,我们以平均每秒完成的指令数来衡量,即 BIPS;其中,GPU 加速部件的执行速度是以 CPU 为基准的等效值,即 CPU 执行速度与 GPU 加速比的乘积.从图中对比结果可以看出,由于程序本身的特征,如计算密集性、分支指令比例等,各程序实际获得的执行性能具有较大的差异.如:计算密集性较高的 Cluster 程序,GPU 达到了非常高的执行性能;而 RPRJ3 程序具有大量非连续的访存操作,而且计算密集性不足以隐藏访存延迟,GPU 仅达到了与 CPU 相当的执行性能.在 SWIM 程序中,CALC1,CALC2 和 CALC3 这 3 个核心程序的计算密集性由高至低.相应地,GPU 的执行性能也随之降低,其变化趋势较 CPU 更为明显,证明了 GPU 更适宜处理高计算密集性应用.由图中 CPU-GPU 的对比结果可知,两类处理器具有明显的应用偏好,因此,应根据程序在异构处理器上的实际执行性能,采用应用感知的功耗管理方式优化异构并行处理性能.此外,对比不同应用程序的执行性能我们发现,不能简单地以程序的绝对 BIPS 值划分功耗系统,应综合考虑并发应用整体效能与公平性划分系统功耗.

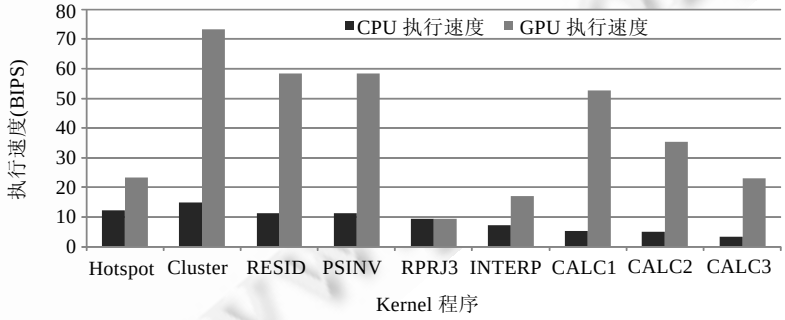


Fig.6 Sustained processing speeds for heterogeneous processors

图 6 异构处理器实际性能对比图

下面我们首先评估应用感知的 HPE 级功耗控制方法.在该实验中,系统每次仅执行单个应用程序.在 CPU 和 GPU 并行执行的情况下,我们无法获得单个处理器的功耗开销.因此,基于以上各程序的性能和功耗开销,实

现本文提出的功耗管理方法.

准确的功耗控制不仅有利于提高系统利用率,而且可以避免产生过高的功耗.图 7 给出了不同功耗约束下系统的实际功耗开销,其中,横坐标表示给定的功耗约束,纵坐标是实际的功耗值.为简化起见,图中只给出了各应用程序中随机选取的一个 Kernel 程序的执行情况.从图中可以看出,本文提出的功耗控制方法可以在不超过功耗约束的前提下,较为准确地逼近给定功耗约束.在上述 4 个 Kernel 程序中,只有 CALC1 程序的实际功耗与约束功耗差距略大,通过分析我们发现,由于物理处理器只支持数个离散的运行频率而且存在上下界约束,当功耗约束为 80%和 90%时恰好无法利用给定的功耗约束,导致功耗浪费现象的发生.

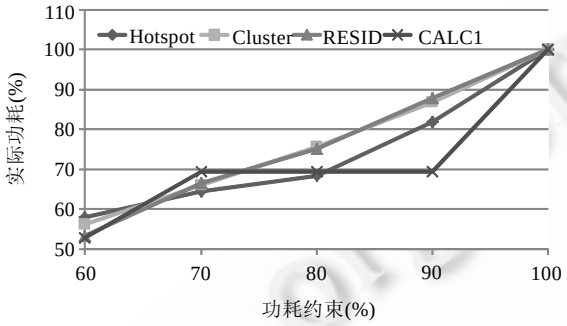


Fig.7 Power control accuracy for Kernel programs

图 7 Kernel 程序功耗控制准确度

除控制准确度外,功耗约束对性能的影响也是衡量功耗控制方法的重要指标.图 8 给出了各程序性能损失随功耗约束的变化情况.在该实验中,我们以各程序在最高频率下并行执行的功耗开销作为基准,评估在功耗约束降低的条件下程序执行时间的变化情况.对于仅由 CPU 执行的计算段,其功耗开销远小于功耗约束,因此,其执行情况不受功耗约束的影响.为了更加清楚地体现优化效果,图中结果仅包含 CPU 和 GPU 并行执行计算段的执行时间.作为对比,我们同时给出了固定任务划分的执行情况,该方法在固定任务划分的情况下仅通过调节处理器运行频率以满足功耗约束,记为 Fixed.与之相应的,本文提出的 PcPar 调度方法通过协调任务划分和频率调节以满足功耗约束.

由图 8 可知,随着系统功耗约束的降低,程序性能损失随之增加.尤其是计算密集型应用 HotSpot,*K*-means 和 MGRID,程序性能受功耗约束影响较为明显;而访存密集型应用 SWIM 受功耗约束的影响相对较小.由于异构处理单元在不同频率下的功耗和性能变化趋势不同,因此,固定的划分策略 Fixed 难以高效开发异构并行处理优势;本文提出的结合任务划分和频率调节的 PcPar 调度方法可以有效地发掘异构并行处理的优势,在相同的功耗约束下获得更高的执行性能.例如,当功耗约束降低为 90%时,本文提出的 PcPar 方法较 Fixed 方法的性能提高达 7.3%.同时,从图 8 可以看出,在少数情况下,PaPar 方法的执行性能低于 Fixed 方法.主要原因是,本文提出的方法是在假设处理器频率连续可调的前提下给出的理论最优结果,而处理器频率离散可调的现状偏离了最优结果;Fixed 方法通过遍历各处理器所有频率组合,在离散空间求解最优频率值,因此在部分情况下会得到性能更优的结果.但是显然,Fixed 方法的复杂度明显高于本文提出的 PcPar 方法,难以适用于未来众核体系结构.即使如此,本文提出的 PcPar 方法在大多数情况下仍能达到更高的执行性能.

下面我们评估系统级功耗控制方法.由于测试平台中仅包含两个 GPU,我们将一个 CPU 核心和一个 GPU 组成 HPE,并将两个独立程序分别映射在两个 HPE 上并发执行.图 9 给出了当系统功耗降低为原最大值 80%的情况下,各程序组合的性能变化情况,其中,First 和 Second 分别表示组合中的第 1 个和第 2 个程序,而 Total 表示程序组合.从图 9 可以看出,大多数应用组合中并发应用之间具有相近的性能损失,并发应用间的平均性能损失差别仅为 5.2%.组合 H-M 和 K-M 中两个应用的性能损失差距略大,通过分析我们发现,HotSpot 和 *K*-means 均是高计算密集型应用,故其执行效能显著高于 MGRID.因此,当这两个应用与 MGRID 组合时将分得更多的功耗配额,并具有更高的执行速度.

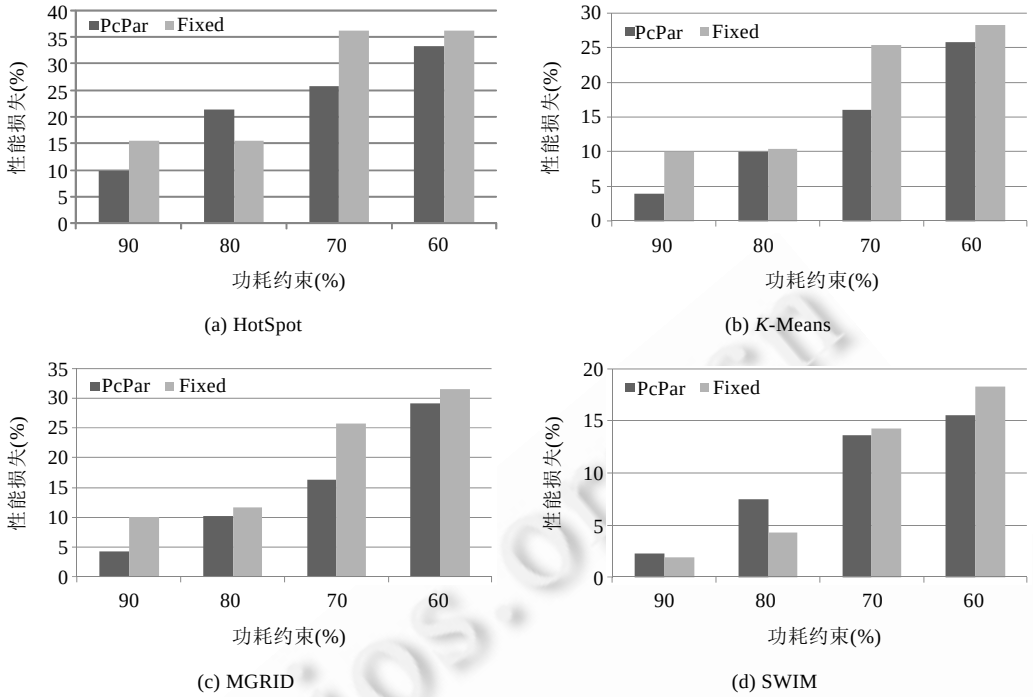


Fig.8 Performance losses under reduced power budget

图 8 性能损失随最大功耗约束的变化情况

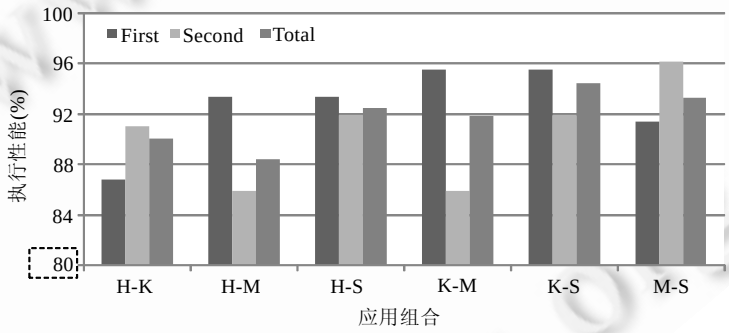


Fig.9 Performance losses for different application mixes with 80% power budget

图 9 系统功耗降低为最大值 80%的情况下各程序组合的性能变化情况

5 相关工作

文献[3]提出了一种基于反馈控制理论的集群功耗管理方法,在总功耗开销一定的条件下,根据不同服务器中处理器的利用率分配总功耗.与文献[3]研究的问题相似,文献[4]提出了一种融合多级功耗管理技术的数据中心功耗管理框架.此外,在微处理器级也有一些最大功耗管理方法的研究.文献[5]提出了一种基于预测的管理策略(MaxBIPS),通过搜索各处理器支持的频率级别,在满足芯片功耗约束的条件下最大化整体吞吐率.文献[6]提出了一种层次式最大功耗管理方法,采用最优提升策略优先将功耗分配给性能收益与功耗开销比值较大的处理器核心.但是,上述两种方法均假设运行在不同处理器核心的应用程序相互独立,难以适用于多线程并行应用环境.文献[7]提出了一种基于控制理论的层次式管理方法,在考虑并发应用间功耗分配的同时,尤其关注同一应

用内不同线程间负载不平衡对功耗分配的影响.然而,已有的功耗管理方法大都面向同构并行系统,未考虑不同计算单元间的速度或功耗差异,使之难以高效地应用于基于加速部件的异构并行系统.

随着以GPU为代表的加速器进入通用计算领域,面向CPU-GPU异构系统的任务调度与功耗优化问题逐渐成为该领域的研究热点.文献[13]根据程序在CPU和GPU上的执行时间和功耗开销,动态地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,仅由特定处理器决定,因此无法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.在文献[14,15]中,Hong 通过对GPU 体系结构的深入剖析,分别建立了 GPU 的性能分析模型和功耗分析模型,对 GPU 相关研究具有重要的指导价值.文献[16]针对 CPU-GPU 异构并行系统建立了一种两阶段能耗优化方法,首先根据异构处理器执行速度划分并行任务已达到负载平衡,然后进一步调节 GPU 上计算核心和存储单元的频率,在不影响执行时间的条件下降低能耗开销.文献[17]以任务图为基础,建立了能耗感知的异构系统任务调度方法,并基于真实的 CPU-GPU 异构系统验证了方法的有效性.已有 CPU-GPU 异构系统功耗优化研究均针对性能约束下的能耗优化问题,然而,随着 GPU 等加速部件性能的不断攀升,功耗约束下的性能优化问题将变得更为严峻和迫切,本文正是针对这类问题来展开研究的.文献[18]提出,通过调节 GPU 核心数量和运行频率的方法以在满足功耗约束的条件下优化芯片吞吐率,该方法只针对 GPU 芯片未能从系统角度优化异构并行系统整体吞吐率的情况.

6 结束语

高功耗已成为制约高性能计算机发展的主要瓶颈之一,近年来,越来越多的研究探索如何在满足系统功耗约束的条件下优化系统整体性能的问题.本文针对基于加速器的异构并行系统,面向多道多线程程序环境,提出了融合两级功耗管理技术的系统功耗管理框架.首先,系统级功耗控制器在给定系统功耗约束的条件下,通过考察各并发应用程序的执行效能分配系统功耗;其次,异构处理引擎功耗控制器采用应用感知的功耗控制策略,通过协调并行任务划分和处理器频率调节技术在满足功耗约束的条件下优化异构并行处理性能.通过 4 个典型的科学计算应用测试,本文提出的功耗控制框架可以较为准确地逼近系统功耗约束,异构并行处理引擎功耗控制器与传统应用透明的管理方法相比,性能提升了 7.3%;系统级功耗控制器可以较好地逼近给定约束值,同时达到了较好的公平性,并发应用间的平均性能损失差别仅为 5.2%.

References:

[1] TOP500 List of the World's Top Supercomputers (in Chinese). http://www.top500.org/lists/2012/06

[2] Mudge T. Power: A first class design constraint for future architectures. IEEE Computer, 2001,34(4):52−58. [doi: 10.1109/2.917539]

[3] Wang X, Chen M. Cluster-Level feedback power control for performance optimization. In: Proc. of the 17th Int'l Conf. on High-Performance Computer Architecture. 2008. 101−110. [doi: 10.1109/HPCA.2008.4658631]

[4] Raghavendra R, Ranganathan P, Talwar V, Wang ZK, Zhu XY. No "power" struggles: Coordinated multi-level power management for the data center. In: Proc. of the 13th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems. New York, 2008. 48−59. http://dl.acm.org/citation.cfm?id=1346289

[5] Isci C, Buyuktosunoglu A, Cher CY, Bose P, Martonosi M. An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget. In: Proc. of the 39th Annual IEEE/ACM Int'l Symp. on Microarchitecture. Washington, 2006. 347−358. [doi: 10.1109/MICRO.2006.8]

[6] Meng K, Joseph R, Dick RP, Shang L. Multi-Optimization power management for chip multiprocessors. In: Proc. of the 15th Int'l Conf. on Parallel Architectures and Compilation Techniques (PACT 2008). New York, 2008. 177−186. http://dl.acm.org/citation.cfm?id=1454141

[7] Ma K, Li X, Chen M, Wang XR. Scalable power control for many-core architectures running multi-threaded applications. In: Proc. of the 38th Annual Int'l Symp. on Computer Architecture. New York, 2011. 449−460. http://dl.acm.org/citation.cfm?id=2000117

[8] Weaver N, Paxson V, Gonzalez JM. The shunt: An FPGA-based accelerator for network intrusion prevention. In: Proc. of the 2007 ACM/SIGDA 15th Int'l Symp. on Field Programmable Gate Arrays. New York, 2007. 199−206. [doi: 10.1145/1216919.1216952]

[9] Winter JA, Albonesi DH, Shoemaker CA. Scalable thread scheduling and global power management for heterogeneous many-core architectures. In: Proc. of the 19th Int'l Conf. on Parallel Architectures and Compilation Techniques. New York, 2010. 29−40. [doi: 10.1145/1854273.1854283]

[10] Intel Corporation. Intel performance counter monitor. 2012. http://software.intel.com/en-us/

[11] Yang CQ, Wang F, Du YF, Chen J, Yi HZ, Lu K. Adaptive optimization for petascale heterogeneous CPU/GPU computing. In: Proc. of the IEEE Int'l Conf. on Cluster Computing. Los Alamitos, 2010. 19−28. [doi: 10.1109/CLUSTER.2010.12]

[12] Che S, Boyer M, Meng JY, Tarjan D, Sheaffer JW, Lee SH, Skadron K. Rodinia: A benchmark suite for heterogeneous computing. In: Proc. of the IEEE Int'l Symp. on Workload Characterization (IISWC 2009). 2009. 44−54. [doi: 10.1109/IISWC.2009.5306797]

[13] Takizawa H, Sato K, Kobayashi H. SPRAT: Runtime processor selection for energy-aware computing. In: Proc. of the IEEE Int'l Conf. on Cluster Computing. 2008. 386−393. [doi: 10.1109/CLUSTR.2008.4663799]

[14] Hong S, Kim H. An analytical model for a GPU architecture with memory-level and thread-level parallelism awareness. In: Proc. of the 36th Int'l Symp. on Computer Architecture (ISCA). 2009. 152−163. [doi: 10.1145/1555815.1555775]

[15] Hong S, Kim H. An integrated GPU power and performance model. In: Proc. of the 37th Annual Int'l Symp. on Computer Architecture (ISCA 2010). New York, 2010. 280−289. [doi: 10.1145/1816038.1815998]

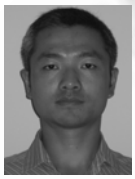
[16] Ma K, Li X, Chen W, Zhang C, Wang XR. GreenGPU: A holistic approach to energy efficiency in gpu-cpu heterogeneous architectures. In: Proc. of the 41st Int'l Conf. on Parallel Processing (ICPP 2012). Pittsburgh, 2012. [doi: 10.1109/ICPP.2012.31]

[17] Liu C, Li J, Huang W, Rubio JC, Speight E, Lin X. Power-Efficient time-sensitive mapping in CPU/GPU heterogeneous systems. In: Proc. of the Int'l Conf. on Parallel Architectures and Compilation Techniques (PACT 2012). 2012. 23−32. [doi: 10.1145/2370816.2370822]

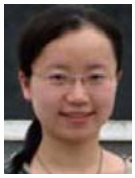
[18] Lee J, Sathisha V, Schulte M, Compton K, Kim NS. Improving throughput of power-constrained GPUs using dynamic voltage/frequency and core scaling. In: Proc. of the 2011 Int'l Conf. on Parallel Architectures and Compilation Techniques (PACT 2011). Washington, 2011. 111−120. [doi: 10.1109/PACT.2011.17]

附中文参考文献:

[1] TOP500 超级计算机排名. http://www.top500.org/lists/2012/06



王桂彬(1981－),男,天津人,博士,助理研究员,CCF 会员,主要研究领域为异构并行体系结构,低功耗编译优化技术.
E-mail: wgbljl@gmail.com



杜静(1979－),女,博士,助理研究员,CCF 会员,主要研究领域为计算机体系结构,高性能编译.
E-mail: jdstarry@yahoo.com.cn

唐滔(1984－),男,博士,助理研究员,主要研究领域为计算机体系结构,高性能编译.
E-mail: tangtao84@nudt.edu.cn