软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(6):1382−1396 [doi: 10.3724/SP.J.1001.2012.04078]
©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

异构并行系统能耗优化分析模型*

王桂彬+, 杨学军, 唐 滔, 徐新海

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

Energy Optimization Model for Heterogeneous Parallel System

WANG Gui-Bin+, YANG Xue-Jun, TANG Tao, XU Xin-Hai

(National Laboratory for 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, Tang T, Xu XH. Energy optimization model for heterogeneous parallel system. *Journal of Software*, 2012,23(6):1382−1396. http://www.jos.org.cn/1000-9825/4078.htm

Abstract: As the processor's power consumption continually increases, power has become the most critical problems during the design and implementation of high performance computer (HPC) system. Nowadays, the heterogeneous system has been one important trend for HPC system. Compared with the traditional homogeneous system, the heterogeneous system has a higher theoretical performance and energy efficiency. However, explointing the potential advantage under the performance constraint is still a challenging problem. This work first establishs the energy optimization model for heterogeneous parallel system via abstracting common applications into the general program model which consists of sequential section and parallel section. Through theoretical analysis, the study conclude the relationship among heterogeneous processors for the minimum energy consumption during the single parallel section and full application (including multiple sections) and provide the corresponding algorithms to decide the operating frequencies under the given performance constraint. Finally, the study evaluates of the proposed model with eight typical applications on a CPU-GPU heterogeneous system.

Key words: heterogeneous system; low-power optimization; task scheduling; dynamic voltage/frequency scaling

摘 要: 随着处理器功耗不断增大,功耗问题逐渐成为高性能计算机系统设计与实现的首要问题.当前,异构系统已成为高性能计算机的发展趋势之一.与传统同构体系结构相比,异构体系结构具有更高的理论峰值性能和能效,但是如何在满足应用性能的条件下充分发掘异构系统的能效优势,仍是一个挑战性问题.通过将应用程序抽象为由串行段和并行段组成的一般程序模型,建立了异构并行系统能耗优化模型.通过分析方法依次给出并行段以及全程序(多程序段)能耗最优时处理器间满足的关系,分别给出了时间约束下能耗最优的处理器频率选择算法.最后,以CPU-GPU 异构系统为平台,通过 8 个典型应用程序验证了方法的有效性.

关键词: 异构系统;低功耗优化;任务调度;动态电压/频率调节

中图法分类号: TP314 **文献标识码**: A

* 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: 2011-02-14; 修改时间: 2011-04-28; 定稿时间: 2011-06-20

© 中国科学院软件研究所 http://www.jos.org.cn

随着处理器功耗不断增大,功耗问题已经成为制约处理器性能提升的主要瓶颈之一[1].过高的功耗不仅增加了芯片的封装和散热成本,还降低芯片的可靠性,影响了高性能计算机系统的可扩展性.以 2010 年 11 月发布的 TOP500[2]前 5 台高性能计算机为例,系统平均功耗达到 3.58MW.因此,IBM 将功耗问题列为未来百万万亿次高性能计算机系统面临的首要问题.

异构并行系统已成为当前高性能计算机系统发展的重要趋势之一,TOP500 前 5 台高性能计算机中有 3 台采用异构体系结构模型[2].异构并行体系结构通过集成通用处理器和高能效(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,有效提高了系统整体能效.TOP500 排名首位的 Tianhe-1A 高性能计算机就是异构系统的典型案例,通过集成通用处理器和图形处理单元(graphics processing units,简称 GPU),系统能效达到 635.15MFLOPS/W.在学术领域,文献[3]对比了传统多核、同构众核和异构众核这 3 种体系结构,分析得出异构体系结构具备更高的峰值性能.文献[4]在此基础上分析得出,异构体系结构不仅具有较高的性能优势,也具有更大的功耗优化空间.

综上所述,与传统同构体系结构相比,异构体系结构具有高性能和高能效的优势,但是如何在满足应用性能的条件下充分发掘异构系统的能效优势,是一个挑战性问题.文献[5]面向同构体系结构研究了程序并行性与性能、并行性与能耗的关系,尤其分析了在加速比一定的条件下如何调节程序中串行段和并行段的处理器运行频率以达到最优能耗.本文试图将以上分析结果拓展到异构体系结构,研究性能约束下异构体系结构能耗最优化问题.本文的主要贡献:

(1) 将应用程序抽象为由串行段和并行段组成的一般程序模型,建立了异构体系结构程序能耗优化模型;

(2) 针对并行段程序,分析得出能耗最优时处理器间满足的关系,基于此,给出了时间约束下能耗最优的处理器频率选择算法,建立了并行段能耗与执行时间的关系;

(3) 基于各程序段能耗与执行时间的关系,将异构体系结构程序能耗优化问题归纳为多元极值问题.为降低计算复杂度,将多个由单一类型处理器完成的程序段(简称同构程序段)的能耗优化问题合并,分析得出同构程序段能耗最优时处理器间满足的关系,建立了同构程序段能耗与执行时间的关系;

(4) 基于 CPU-GPU 异构并行系统,通过 8 个典型应用程序验证了本文提出的异构系统能耗优化模型可以有效降低系统能耗,提高系统能效.

本文第 1 节介绍异构体系结构能耗优化模型及相关概念.第 2 节讨论并行段的能耗优化问题.第 3 节分析同构程序段的能耗优化问题,并将全程序功耗优化问题归纳为多元极值问题.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研究进展.第 6 节总结本文工作,并给出今后的研究问题.

1 问题提出

本节首先给出文中对异构并行系统的假设.异构并行系统中可能包含 CPU,GPU 或 CELL 等多种类型的计算资源,本文将各种类型的计算资源统称为处理器.不失一般性,假设系统由 m 类处理器组成,记为 $R=\{r_0,\ldots,r_{m-1}\}$,其中,第 $j(0\leqslant j\leqslant m-1)$类处理器 r_j 的数量记为 N_j,其在最高频率下的功耗和速度分别记为 P_j 和 V_j,其中速度指处理器单位时间内完成任务量.

不失一般性,假设程序由串行段和并行段两类程序段组成,根据程序段的并行性将程序分为 n 段,记为 $S=\{s_0,\ldots,s_{n-1}\}$,其中,s_i 表示第 i 个程序段的任务量.图 1 给出了文中程序模型的示意图.

本文的能耗优化过程是在不改变程序在异构多处理器上映射关系的前提下展开的,程序段与处理器的映射关系记为 $F:s_i\rightarrow R_i$(其中,$R_i\subseteq R$).如果 s_i 为串行段,则集合 R_i 中仅包含一种类型的处理器,此时由该类型处理器中的单个处理器完成;如果 s_i 为并行段,则由 R_i 中所有类型的处理器并行完成.以图 1 中的示例程序为例,其中,串行段分别由指定的处理器独立完成,并行段可由两类处理器并行完成.本文不关注并行段的具体并行化过程,同时假设并行段可连续划分.在实际应用中,并行段应按照某种粒度实现并行化,以 OpenMP 编程模型为例,其并行段以并行循环迭代为粒度调度到多处理器上执行.因此,应将本文给出的连续解离散化取得特定编程模型支持的并行粒度.同样以 OpenMP 编程模型为例,其循环迭代空间为整数空间,因此,应将本文给出的理论最优解取

© 中国科学院软件研究所 http://www.jos.org.cn

整得到合理结果,该过程可通过编译器自动实现.

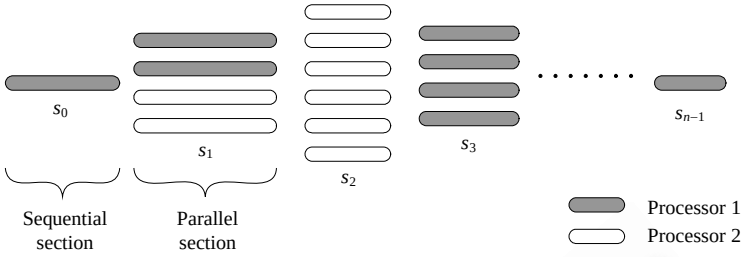


Fig.1 An example program

图 1 程序模型示意图

记第 i 个程序段的执行时间为 t_i,能量开销为 e_i.按照物理学定律,能耗是处理器功耗与执行时间的乘积,因此对于第 i 个程序段,能耗 e_i 可表示为

$$e_i = \sum_{r_j \in R_i} N_j p_j t_i,$$

其中,$p_j(0 \leqslant p_j \leqslant P_j)$表示第 j 类处理器的实际动态功耗.由文献[6]可知,对于同构处理器系统,当处理器以相同的功耗完成等量的计算任务时总能耗达到最优,因此我们用 p_j 表示第 j 类处理器的功耗.

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 $p=ACV^2f$,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.而运行频率与核心电压的关系为 $f = K\frac{(V-V_T)^\gamma}{V}$ $(1 \leqslant \gamma \leqslant 2)$,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,因此频率与电压的关系近似表示为 $f=KV^{\gamma-1}$.

可知,动态功耗与频率的关系可表示为 $p=ACKf^\alpha$,其中,$\alpha = \frac{\gamma+1}{\gamma-1}$.由该式可知,处理器动态功耗与频率的 α 次方成正比关系.在异构处理器系统中,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,即对于第 j 类处理器有 $p_j = f_j^\alpha P_j$,其中,f_j 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.因此,第 i 个程序段的能耗 e_i 为

$$e_i = \sum_{r_j \in R_i} N_j P_j f_j^\alpha t_i \tag{1}$$

考虑集合 R_i 中的处理器必须在约束时间 t_i 内完成任务 s_i,因此处理器运行频率应满足如下不等式:

$$\sum_{r_j \in R_i} N_j V_j f_j \geqslant \frac{s_i}{t_i} \tag{2}$$

综合公式(1)和公式(2)可知,第 i 个程序段的能耗是关于执行时间 t_i 和处理器运行频率 f_j 的函数,记为 $e_i(t_i,f_i)$,其中,$r_j \in R_i$.

本文研究的问题是针对由多个程序段组成的程序模型,求解在给定执行时间 T 的约束下使全程序总能耗达到最小.首先给出该问题的形式化描述:

$$\begin{cases} \min . E = \sum_{i=0}^{n-1} e_i(t_i, f_j), r_j \in R_i \\ \text{s.t.} \sum_{i=0}^{n-1} t_i \leqslant T \\ t_i \geqslant \frac{s_i}{V_j}, & \text{if } s_i \text{ is sequential section or} \\ t_i \geqslant \frac{s_i}{\sum_{r_j \in R_i} N_j V_j}, & \text{if } s_i \text{ is parallell section} \end{cases} \tag{3}$$

© 中国科学院软件研究所 http://www.jos.org.cn

其中,对于任意程序段 s_i 的时间约束 t_i 的分析是:如果第 i 个程序段 s_i 为串行段,则该程序段仅由一个处理器完成,此时执行时间满足 $t_i \geqslant \frac{s_i}{V_j}$ ($\{r_j\}=R_i$);如果 s_i 为并行段,则该程序段由集合 R_i 中所有处理器并行完成,此时执行时间 t_i 满足 $t_i \geqslant s_i \Big/ \sum_{r_j \in R_i} N_j V_j$.

由公式(3)可知,能耗最优问题原则上可以分为两步求解,即程序段内局部能耗最优和全程序整体能耗最优.第 1 个子问题是求解程序段内处理器最优能耗与执行时间的关系,即在给定时间约束 t_i 的条件下确定各类处理器的运行频率 f_j.文中第 2 节首先分析了异构多处理器达到能耗最优时处理器间满足的关系,并基于此给出了最优能耗与执行时间的关系 $e_i(t_i)$.

第 2 个问题是在程序段内能耗最优的基础上分配不同程序段的执行时间,在满足时间约束的条件下达到全程序能耗最优.由公式(1)可知,增大某程序段的执行时间可以降低该程序段的能量消耗;但是在总执行时间一定的条件下,影响了其他程序段的能耗优化空间.因此应综合考虑所有程序段的能耗开销,分配各程序段的执行时间.由公式(3)可知,该问题是一个 n 元极值问题,其最优解的计算开销较大.为简化问题求解,我们将仅由单一类型处理器完成的程序段抽取并构成同构程序段.以图 1 为例,并行段 s_2 和 s_3 都仅由单一类型的处理器完成,而串行段 s_0 和 s_{n-1} 显然仅由单一类型的处理器完成,因此示例程序中同构程序段为 $\{s_0,s_2,s_3,s_{n-1}\}$.我们将同构程序段的时间分配问题看做子问题,文中第 3 节通过分析得出同构程序段最优能耗与时间的关系,进而得到简化后的多元极值问题.

2 并行段能耗优化分析

根据问题假设,并行段可由多个同构或异构处理器并行完成.文献[6]研究了同构多处理器完成并行计算任务的最优能耗问题,分析得出,在同构多处理器系统中,当所有处理器以相同的运行频率完成等量的计算任务时,总能耗达到最优.本节首先分析在异构多处理器系统中,当总能耗达到最优时,处理器间将应满足怎样的关系,定理 1 分析了该问题.

定理 1(异构处理器并行执行能效平衡定理). 假设第 i 个程序段 s_i 由异构多处理器集合 R_i 并行完成,根据执行时间 t_i 的大小,系统能耗达到最优时,处理器间满足如下关系:

1. 如果 $t_i \geqslant \frac{s_i}{\rho}\psi^{1/(\alpha-1)}$,则所有处理器的能效相等,即 $\frac{v_i^j}{p_i^j}=\frac{v_i^k}{p_i^k}$ ($r_j,r_k \in R_i$);
2. 如果 $\frac{s_i}{\sum_{r_j \in R_i} N_j V_j} \leqslant t_i < \frac{s_i}{\rho}\psi^{1/(\alpha-1)}$,则必有部分处理器运行在最高频率,且剩余处理器集合中各处理器的能效相等.

其中,$\rho=\sum_{r_j \in R_i} N_j \beta_j^{-1}$,$\beta_j=(P_j/V_j^{\alpha})^{1/(\alpha-1)}$,$\psi=\max\{V_j/P_j | r_j \in R_i\}$,$v_i^j$ 和 p_i^j 分别为第 j 类处理器在第 i 个程序段的实际执行速度和动态功耗.

证明:记第 j 类处理器 r_j 完成的任务总量为 $s_i^j (0<s_i^j<s_i)$.可知处理器动态功耗:

$$p_i^j = P_j (f_i^j)^{\alpha} = P_j \left(\frac{s_i^j}{V_j N_j t_i}\right)^{\alpha} = \frac{\beta_j^{\alpha-1}(s_i^j)^{\alpha}}{N_j^{\alpha} t_i^{\alpha}},$$

其中,$\beta_j=(P_j/V_j^{\alpha})^{1/(\alpha-1)}$,则总能耗为

$$e_i = \sum_{r_j \in R_i} p_i^j N_j t_i = \sum_{r_j \in R_i} \frac{\beta_j^{\alpha-1}(s_i^j)^{\alpha}}{t_i^{\alpha-1} N_j^{\alpha-1}} = \frac{1}{t_i^{\alpha-1}} \sum_{r_j \in R_i} \frac{\beta_j^{\alpha-1}(s_i^j)^{\alpha}}{N_j^{\alpha-1}}.$$

考虑程序段总任务约束 F: $\sum_{r_j \in R_i} s_i^j - s_i = 0$.通过拉格朗日乘子法求极值,令 $\frac{\partial E}{\partial s_i^j} = \kappa \frac{\partial F}{\partial s_i^j}$,可得

© 中国科学院软件研究所 http://www.jos.org.cn

$$(s_i^j)^{\alpha-1}=\frac{\kappa t_i^{\alpha-1}N_j^{\alpha-1}}{\alpha\beta_j^{\alpha-1}}.$$

带入功耗表达式中,可得

$$p_i^j=\frac{\beta_j^{\alpha-1}(s_i^j)^{\alpha}}{N_j^{\alpha}t_i^{\alpha}}=\frac{\kappa s_i^j}{\alpha t_i N_j},$$

即

$$\frac{s_i^j}{N_j t_i p_i^j}=\frac{v_i^j}{p_i^j}=\frac{\alpha}{\kappa},$$

其中, $v_i^j=\frac{s_i^j}{N_j t_i}$.可知,当总能耗达到最优时,所有处理器的能效相等,即 $\frac{v_i^j}{p_i^j}=\frac{v_i^k}{p_i^k}$ $(\forall r_j,r_k\in R_i)$.

此时,第 j 类处理器分得的任务量 $s_i^j=\frac{N_j\beta_j^{-1}}{\rho}s_i$.

需要注意的是,上述分析中必须满足$\forall r_j\in R_i$, $f_i^j\leqslant 1$,即

$$f_i^j=\frac{s_i^j}{V_j N_j t_i}=\frac{N_j\beta_j^{-1}}{\sum_{r_k\in R_i}N_k\beta_k^{-1}}\frac{s_i}{V_j N_j}=\frac{s_i}{\rho t_i\beta_j V_j}\leqslant 1,$$

其中, $\rho=\sum_{r_k\in R_i}N_k\beta_k^{-1}$.即第 i 个程序段的执行时间应满足不等式 $t_i\geqslant\frac{s_i}{\rho\beta_j V_j}=\frac{s_i}{\rho}\left(\frac{V_j}{P_j}\right)^{1/(\alpha-1)}$.

综上所述,当执行时间 T 满足 $t_i\geqslant\frac{s_i}{\rho}\psi^{1/(\alpha-1)}$,所有处理器的能效相等,其中,$\psi=\max\{V_j/P_j|r_j\in R_i\}$.

下面分析当 $t_i<\frac{s_i}{\rho}\psi^{1/(\alpha-1)}$ 时,处理器间的关系.不难证明,t_i 的下界是 $s_i\Big/\sum_{r_j\in R_i}N_jV_j$.

如果 $s_i\Big/\sum_{r_j\in R_i}N_jV_j\leqslant t_i<\frac{s_i}{\rho}\psi^{1/(\alpha-1)}$,则至少有一类处理器 j 满足 $f_j>1$.即分配给第 j 个处理器的任务量超过其在约束时间内可以完成的最大计算量.由于能耗 e_i 是关于 s_i^j 的凸函数,因此当 $s_i^j=V_jN_jt_i$ 时,总能耗达到最优,即 $f_i^j=1$;此后,再将剩余任务 $s_i-s_i^j$ 分配给其余 n–1 类处理器,如果此时满足条件 1 中的约束,则处理器必满足能效平衡关系;否则,重复上述过程,直至满足条件 1 中的约束. □

由定理 1 的证明过程可知,当执行时间 $t_i\geqslant\frac{s_i}{\rho}\psi^{1/(\alpha-1)}$ 时,所有处理器的运行频率都小于其最大运行频率,此时,第 j 类处理器分得的任务量 $s_i^j=\frac{N_j\beta_j^{-1}}{\rho}s_i$,可得处理器的相对运行频率为 $f_i^j=\frac{s_i^j}{V_jN_jt_i}=\frac{s_i}{\rho V_j\beta_j t_i}$.由处理器频率表达式可知,参数 s_i,ρ,V_j 和 β_j 都是与程序或体系结构相关的常量.如果执行时间进一步放松为 $t_i'(t_i'>t_i)$ 时,能耗最优的处理器频率为 $f_i^{j'}=\frac{s_i}{\rho V_j\beta_j t_i'}=\frac{t_i}{t_i'}f_i^j$.由此可知,当执行时间进一步放松为 t_i' 时,所有处理器按照相同的比例 $\frac{t_i}{t_i'}$ 缩小即可达到最优能耗.

根据执行时间 t_i 的大小,可以将并行任务调度问题看做多阶段分配问题.图 2 给出时间约束下能耗最优的任务调度与频率选择算法 EoPT.如果在算法执行前首先根据处理器的最大功耗进行排序,则算法复杂度为 $O(m_i\log m_i)$,其中,$m_i=|R_i|$.基于算法 EoPT,可以得出任意并行段最优能耗值与执行时间的关系 $e_i(t_i)$.

© 中国科学院软件研究所 http://www.jos.org.cn

Algorithm EoPT. Energy-Optimal Parallel Task Scheduler for Heterogeneous Multiprocessors.
Input: Execution Time t_i, Task Requirement s_i, Processor List $R_i=\{r_j\}$, $r_j=\langle P_j,V_j,N_j\rangle$;
Output: Work distribution $\{s_i^j\}$ and operating frequency $\{f_i^j\}$ for each processor r_j.

Set $\psi=\max\{V_j/P_j|r_j\in R_i\}$ and $\rho=\sum_{r_j\in R_i}N_j\beta_j^{-1}$, while $\beta_j=(P_j/V_j^{\alpha})^{1/(\alpha-1)}$

While $t_i\geqslant\frac{s_i}{\rho}\psi^{1/(\alpha-1)}$

find the k processor achieving $V_k/P_k=\max\{V_j/P_j|r_j\in R_i\}$
set $f_i^k=1$, $s_i^k=V_kN_kt_i$, $s_i=s_i-s_i^k$
remove processor k from R_i, re-compute ρ and ψ.
end

for each $r_j\in R_i$, set $f_i^j=\frac{s_i}{\rho t_i\beta_jV_j}$, $s_i^j=\frac{N_j\beta_j^{-1}}{\rho}s_i$

Fig.2 Performance constrained energy-optimal frequency scaling algorithm

图 2 时间约束下并行段能耗最优的处理器频率选择算法

定理 1 中假设任务 s_i 可以完全并行,而在程序并行化的过程中,不可避免地会出现对共享数据的操作.为保证数据一致性,对共享数据的操作必须包含在临界区中.

下面分析临界区同步对功耗优化的影响,记并行段 s_i 中临界区操作的比例为 σ_i.本节的分析基于文献[7]中对临界区的假设,即假设线程内临界区的出现概率服从均匀分布;不同线程间临界区的出现概率完全独立,且单位时间内竞争临界区的线程数量服从二项式分布.

定理 2(临界区对系统能耗优化的影响). 在执行时间一定的条件下,与无临界区的情况相比,临界区操作不会改变能耗最优的任务划分结果,但是会增大处理器的运行频率,进而增大总能耗开销;临界区操作使处理器运行频率增加了 $\sigma_i^2c_in_i$ 倍,即 $f_i^j=\frac{s_i}{\beta_jV_j\rho t_i}(1+\sigma_i^2c_in_i)$ $(r_j\in R_i)$,其中,c_i 为临界区冲突概率, $n_i=\sum_{r_j\in R_i}N_j$.

证明:首先给出相关符号说明,c_i 为临界区冲突概率,即不同临界区竞争同一个锁的概率.如果并行段中所有临界区只使用一个锁,则 c_i=1.

根据文献[7]建立的临界区冲突模型,处理器在临界区的执行时间由本地临界区执行时间和临界区等待时间组成,而等待时间与其他处理器的临界区执行时间、临界区执行概率和冲突概率成正比关系[7].

第 j 类处理器的临界区执行时间为

$$tc_i^j=\frac{\sigma_is_i^j}{f_i^jV_jN_j}+\left((N_j-1)\frac{\sigma_is_i^j}{f_i^jV_jN_j}+\sum_{r_k\in R_i-\{r_j\}}N_k\frac{\sigma_is_i^k}{f_i^kV_kN_k}\right)\sigma_ic_i,$$

其中,第 2 项表示第 j 类处理器的平均等待时间.

考虑 $\frac{s_i^j\sigma_i^2c_i}{f_i^jV_jN_j}$ 占 tc_i^j 的比例较小,因此对上式取近似可得, $tc_i^j=\frac{\sigma_is_i^j}{f_i^jV_jN_j}+\sum_{r_k\in R_i}\frac{s_i^k}{f_i^kV_k}\sigma_i^2c_i$.

在考虑临界区开销的情况下,第 j 类处理器的执行时间为

$$t_i^j=tnc_i^j+tc_i^j=\frac{(1-\sigma_i)s_i^j}{f_i^jV_jN_j}+\frac{\sigma_is_i^j}{f_i^jV_jN_j}+\sum_{r_k\in R_i}\frac{s_i^k}{f_i^kV_k}\sigma_i^2c_i=\frac{s_i^j}{f_i^jV_jN_j}+\sum_{r_k\in R_i}\frac{s_i^k}{f_i^kV_k}\sigma_i^2c_i.$$

由于 $\forall r_j,r_k\in R_i,t_i^j=t_i^k=t_i$,可得 $\frac{s_i^j}{s_i^k}=\frac{f_i^jV_jN_j}{f_i^kV_kN_k}$;再由 $\sum_{r_j\in R_i}s_i^j=s_i$,可得 $s_i^j=\frac{f_i^jV_jN_j}{\sum_{r_k\in R_i}f_i^kV_kN_k}s_i$,故有

$$t_i=\frac{s_i^j}{f_i^jV_jN_j}+\sum_{r_k\in R_i}\frac{s_i^k}{f_i^kV_k}\sigma_i^2c_i=\frac{1+\sigma_i^2c_in_i}{\sum_{r_k\in R_i}f_i^kV_kN_k}s_i,$$

© 中国科学院软件研究所 http://www.jos.org.cn

其中,$n_i=\sum_{r_k\in R_i}N_k$.记 $K=\frac{1+\sigma_i^2c_in_i}{t_i}s_i$,即 $\sum_{r_k\in R_i}f_i^kV_kN_k=K$.

阻塞方式是当前临界区原语的主要实现方式,其实现机制是将等待临界区操作的线程置于挂起状态,直至被操作系统唤醒并进入临界区执行.该方式的优点是减少处于临界区等待状态的线程对处理器资源的竞争,基于此,本文假设等待过程中不消耗动态功耗,后文第 4.3 节的实验验证了这样假设.此时,并行段中动态能耗为

$$e_i=\sum_{r_j\in R_i}N_jP_j(f_i^j)^\alpha t_i^j=\sum_{r_j\in R_i}N_j\frac{s_i^j}{f_i^jV_jN_j}P_j(f_i^j)^\alpha=\sum_{r_j\in R_i}P_jN_j(f_i^j)^\alpha\frac{s_i}{K}.$$

与定理 1 的证明过程相似,联合约束条件 $F:\sum_{r_j\in R_i}f_i^jV_jN_j=K$,由拉格朗日乘子法解得

$$\frac{v_i^j}{p_i^j}=\frac{v_i^k}{p_i^k}\ (\forall r_j,r_k\in R_i).$$

可以看出,在有临界区的并行段中处理器间依然满足能效平衡定理.进而求得第 j 类处理器的运行频率 $f_i^j=\frac{s_i}{\rho t_i\beta_jV_j}(1+\sigma_i^2c_in_i)$.

对比定理 1 的结论可知,在包含临界区操作的并行段中,最优能耗下频率的增加量与 $\sigma_i^2c_in_i$ 呈线性关系.

将上式带入第 j 类处理器的任务表达式中可得,$s_i^j=\frac{f_i^jV_jN_j}{\sum_{r_k\in R_i}f_i^kV_kN_k}s_i=\frac{N_j\beta_j^{-1}}{\rho}s_i$.对比定理 1 中的任务划分结果,可知临界区的出现不会影响最优任务划分结果. □

由定理 2 可知,与无临界区的情况相比,在相同的执行时间约束下,处理器运行频率变为原来的 $1+\sigma_i^2c_in_i$ 倍;而由定理 2 的证明过程可知,此时有效计算时间(即排除临界区等待时间)变为原来的 $1/(1+\sigma_i^2c_in_i)$,因此处理器总能耗增大为原来的 $(1+\sigma_i^2c_in_i)^{\alpha-1}$ 倍.可此可见,缩小 c_i 可以有效降低能耗开销.因此,使用细粒度锁机制,在不改变程序语义的情况下,将临界区拆分为由不同锁变量维护的多个临界区,也是降低能耗的有效方法.

3 全程序能耗优化分析

第 2 节分析了异构多处理器执行并行段时总能耗 e_i 与执行时间 t_i 的关系,而同构多处理器执行并行段的情况可以看做是前者特例.因此,我们可以得出任意程序段能耗与执行时间的关系函数.基于该结果,本节分析在给定总执行时间的条件下如何分配各程序段的执行时间 t_i.

由第 1 节的分析可知,如果将所有程序段的能耗关于执行时间的函数带入公式(3),可以将不同程序段的时间分配问题看做多元极值问题,该问题复杂度随程序段数量的增大而增大.为简化问题求解,我们试图将所有仅由单一类型处理器完成的程序段的时间分配问题合并为一个子问题,简称同构程序段.在同构程序段中,各程序段都只由单一类型的处理器完成,而不同的程序段可以由不同类型的处理器完成.

定理 3 分析了同构程序段总能耗达到最优时处理器间满足的关系.记同构程序段为 S_{syn},其执行时间为 T_{syn}.

定理 3(同构程序段功耗平衡定理). 根据总执行时间 T_{syn} 的大小,当系统达到能耗最优时,处理器间满足如下关系:

1. 如果 $T_{syn}\geqslant\frac{\tau}{\pi^{1/\alpha}}$,则各程序段间处理器总功耗相等,即满足 $\forall s_i,s_k\in S_{syn},N_ip_i=N_kp_k$;
2. 如果 $\sum_{s_i\in S_{syn}}\frac{s_i}{N_jV_j}\leqslant T_{syn}<\frac{\tau}{\pi^{1/\alpha}}$,则必有部分程序段中的处理器运行在最高频率,且其余程序段中处理器的总功耗相等.

其中,$\pi=\min\{P_jN_j|s_i\in S_{syn}\}$,$\tau=\sum_{s_i\in S_{syn}}s_i/\omega_j$,$\omega_j=(N_j/\beta_j)^{(\alpha-1)/\alpha}$,$\beta_j=(P_j/V_j^\alpha)^{\frac{1}{\alpha-1}}$.

© 中国科学院软件研究所 http://www.jos.org.cn

证明:由文献[6]可知,在同构多处理器系统中,当总能耗达到最优时,所有处理器一定运行在相同频率下,即具有相同的功耗.因此,用 p_i 表示第 i 个程序段中处理器功耗.则功耗为

$$p_i = P_j f_i^{\alpha} = P_j\left(\frac{s_i}{N_j V_j t_i}\right)^{\alpha} = \frac{s_i^{\alpha}\beta_j^{\alpha-1}}{t_i^{\alpha}N_j^{\alpha}},$$

其中,$\beta_j = (P_j/V_j^{\alpha})^{\frac{1}{\alpha-1}}$,则能耗为

$$E_{syn} = \sum_{s_i\in S_{syn}} N_i p_i t_i = \sum_{s_i\in S_{syn}} \frac{s_i^{\alpha}\beta_j^{\alpha-1}}{t_i^{\alpha-1}N_j^{\alpha-1}}.$$

由 $T_{syn} = \sum_{s_i\in S_{syn}} t_i$ 可得约束条件 F: $\sum_{s_i\in S_{syn}} t_i - T_{syn} = 0$.

通过拉格朗日乘子法求解最优能耗值,令 $\frac{\partial E}{\partial t_i} = \kappa\frac{\partial F}{\partial t_i}$ (κ为拉格朗日乘子),可得 $\frac{s_i^{\alpha}\beta_j^{\alpha-1}}{t_i^{\alpha}N_j^{\alpha-1}} = \frac{\kappa}{1-\alpha}$;而程序段 s_i 中所有处理器的总功耗为 $N_j p_i = \frac{s_i^{\alpha}\beta_j^{\alpha-1}}{t_i^{\alpha}N_j^{\alpha-1}} = \frac{\kappa}{1-\alpha}$.因此在能耗最优的情况下,对于$\forall s_i,s_k\in S_{syn}$,均有 $N_ip_i=N_kp_k$.即不同程序段间处理器动态功耗总和相等.

需要注意的是,上式成立的条件是所有处理器的运行频率小于其最大值,即 $f_i\leqslant 1$.下面求解上述约束.

通过拉格朗日乘子法,可得 $t_i = \left(\frac{1-\alpha}{\lambda}\right)^{1/\alpha}\frac{s_i}{(N_j/\beta_j)^{(\alpha-1)/\alpha}} = \left(\frac{1-\alpha}{\lambda}\right)^{1/\alpha}\frac{s_i}{\omega_j}$,其中,$\omega_j=(N_j/\beta_j)^{(\alpha-1)/\alpha}$.

由 $T_{syn} = \sum_{s_i\in S_{syn}} t_i$,可知 $t_i = \frac{s_i}{\tau\omega_j}T_{syn}$,其中,$\tau = \sum_{s_i\in S_{syn}} s_i/\omega_j$.则第 i 个程序段中的处理器运行频率为 $f_i = \frac{s_i}{t_iN_jV_j} = \frac{\omega_j\tau}{N_jV_jT_{syn}}\leqslant 1$,即 $T_{syn}\geqslant\frac{\omega_j\tau}{N_jV_j} = \frac{\tau}{(N_iP_j)^{1/\alpha}}$ ($\forall s_i\in S_{syn}$).记 $\pi=\min\{P_jN_j|s_i\in S_{syn}\}$,则执行时间 T_{syn} 需满足以下不等式: $T_{syn}\geqslant\frac{\tau}{\pi^{1/\alpha}}$.

综上所述,当执行时间 T_{syn} 满足 $T_{syn}\geqslant\frac{\tau}{\pi^{1/\alpha}}$ 时,完成不同任务的处理器动态功耗总和相等.

不难证明,执行时间 T_{syn} 的下界是 $\sum_{s_i\in S_{syn}}\frac{s_i}{N_jV_j}$.下面分析当 $\sum_{s_i\in S_{syn}}\frac{s_i}{N_jV_j}\leqslant T_{syn}<\frac{\tau}{\pi^{1/\alpha}}$ 时,能耗最优时处理器间的功耗关系.记动态功耗和为 π 的程序段集合 $S_{syn}^{(1)} = \{s_i\,|\,P_jN_j=\pi, s_i\in S_{syn}\}$.记

$$\tau^{(2)} = \tau - \sum_{s_i\in S_{syn}^{(1)}}\frac{s_i}{\omega_j}, \pi^{(2)} = \min\{P_jN_j\,|\,s_i\in S_{syn}-S_{syn}^{(1)}\}.$$

当 $\frac{\tau^{(2)}}{(\pi^{(2)})^{1/\alpha}}\leqslant T_{syn}<\frac{\tau}{\pi^{1/\alpha}}$ 时,程序段子集 $S_{syn}^{(1)}$ 中所有处理器相对运行频率 f_i 均大于 1.那么,在该情况下,系统能耗最优时 $S_{syn}^{(1)}$ 中处理器是否应该运行在最高频率.由于能耗 E 是关于 t_i 的凸函数,因此可以证明,上述结论是肯定的.即当 $T<\frac{\tau}{\pi^{1/\alpha}}$ 时,完成程序段子集 $S_{syn}^{(1)}$ 中的所有处理器应运行在最高频率.

此时,我们可以将原问题变为系统由 $m^{(2)}(m^{(2)}=|S_{syn}|-|S_{syn}^{(1)}|)$ 类处理器组成,且约束时间为

$$T_{syn}^{(2)} = T_{syn} - \sum_{s_i\in S_{syn}^{(2)}}\frac{s_i}{N_jV_j}.$$

当 $T^{(2)}\geqslant\frac{\tau^{(2)}}{(\pi^{(2)})^{1/\alpha}}$ 时,在能耗最优的情况下,剩余的 $m^{(2)}$ 类处理器的功耗必然相等.以此类推,我们可以根据

© 中国科学院软件研究所 http://www.jos.org.cn

执行时间 T 的大小分为多段,在每个时间段中,未运行在最高频率的处理器集合的总功耗相等. □

由定理 3 可知,时间约束下能耗最优的处理器频率选择问题只与异构系统的处理器类型数有关,而与程序段的数量无关.因此,对于由 m 类处理器组成的异构系统,根据执行时间 T_{syn} 的大小,可以将处理器频率选择问题看做 m 元变量的优化问题,通过反复应用定理 3 的结论即可求解该问题.图 3 给出同构程序段的最优频率选择算法.

Algorithm EoMT. Energy-Optimal Multi-task Scheduler for Heterogeneous Multiprocessors.
Input: Scheduling Length T_{syn}, Homogeneous Task Set S_{syn},,
Task Mapping Relationship F: $s_i \to R_i$, $r_j=\langle P_j,V_j,N_j\rangle$, $\{r_j\}=R_i$;
Output: Operating frequency for each processor r_j.

Set $\pi=\min\{P_jN_j|s_i\in S_{syn}\}$ and $\tau=\sum_{s_i\in S_{syn}} s_i/\omega_j$, while $\omega_j=(N_j/\beta_j)^{(\alpha-1)/\alpha}$, $\beta_j=(P_j/V_j^{\alpha})^{\frac{1}{\alpha-1}}$

While $T_{syn}>\frac{\tau}{\pi^{1/\alpha}}$

find the subset of the task $S_{sub}=\{r_i|P_jN_j=\pi\}$

for all the task $s_i\in S_{sub}$, set $f_j=1$, $T_{syn}=T_{syn}-\sum_{s_i\in R_{sub}}\frac{s_i}{V_jN_j}$

set $S_{syn}=S_{syn}-S_{sub}$, re-compute π and τ.

end

for all the task $s_i\in S_{syn}$, set $f_j=\frac{\omega_j\tau}{N_jV_jT_{syn}}$

Fig.3 Energy-Optimal frequency scaling algorithm for homogeneous program sections

图 3 能耗最优的同构程序段频率选择算法

算法 EoMT 按照各程序段处理器总功耗递增的顺序,依次排除违反频率约束的分配结果;同时移出达到最高运行频率的处理器,直至剩余程序段集合的执行时间满足时间约束.如果首先按照程序段中处理器总功耗进行排序,则算法复杂度为 $O(m\log m)$.基于算法 EoMT,可以得出同构程序段最优能耗与执行时间的关系 $E_{syn}(T_{syn})$.

基于定理 1~定理 3,我们分别建立并行段以及同构程序段中能耗与时间的关系,因此尚未解决的问题是如何在这两类程序段间分配约束时间.我们将该问题归纳为一般多元极值问题,显然,同构程序段的数量为 1.记剩余并行段的数量为 h,因此公式(3)中的优化问题可以进一步描述为

$$\begin{cases}\min.\ E=\sum_{i=0}^{h}e_i(t_i)\\ \text{s.t.}\ \sum_{i=0}^{h}t_i\leqslant T\\ t_i\geqslant\sum_{s_i\in S_{syn}}\frac{s_i}{n_jV_j},\ \text{if}\ s_i\in S_{syn}, n_j=1\ \text{or}\ N_j\\ t_i\geqslant\frac{s_i}{\sum_{r_j\in R_i}N_jV_j},\ \text{otherwise}\end{cases}\tag{4}$$

其中,串行段只可能由一类处理器完成,因此将其归纳为同构程序段中.如公式(4)所示,根据程序段的并行性,n_j 可以取 1 或 N_j.

4 实验与评测

本节依次介绍实验中的软、硬件平台和测试用例,最后给出了实验结果和详细分析.

4.1 测试平台

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.在该异构系统

© 中国科学院软件研究所 http://www.jos.org.cn

中,CPU 和 GPU 分别具有独立的存储空间,且通过 PCI-E 总线相连实现数据通信操作.总线标准为 PCI-E16,最大单向通信带宽为 8GB/s,具体软硬件参数见表 1.

Table 1 Experimental testbed

表 1 测试平台介绍

Processor	Core I7 920 CPU	4870×2 GPU
Core frequency (GHz)	2.67, 2.4, 2.0, 1.6 (0~3)	0.75, 0.65, 0.55 (0~2)
Memory frequency (GHz)	1.33 (DDR3)	0.9 (GDDR5)
Cache	L1 I32KB, D32KB, L2 256KB, L3 8MB	—
Memory space	8 GB	1GB
Compiler	ifort/icc v11.1, gcc v4.2.1	AMD stream/APP SDK
Operating system	OpenSUSE v10.3 x86_64, ACPI enabled	
System bus	PCI-E16, peak bandwidth 8GB/s per direction	

目前,主流通用处理器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.本文基于 OpenSUSE 系统提供的 ACPI 接口,可以动态访问和调节 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.本文通过 AMD 公司提供的 ADL 库(AMD Display Library)获取 GPU 芯片支持的运行级别,并完成 DVFS 操作.

本文通过外接 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.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.

4.2 测试用例

本文选取了 8 个典型科学计算应用程序,见表 2.表中前 6 个应用选自 ATI Stream SDK2.2,且均通过 OpenCL 语言实现.OpenCL 语言同时支持 CPU 和 GPU 平台,因此可以较为公平地比较不同平台的执行效率.目前,ATI Stream SDK 中的应用程序往往仅包含一个 Kernel 程序,为了测试本文提出的优化算法在多 Kernel 程序中的优化效果,我们选择 Swim 和 Mgrid 两个测试程序.Swim 和 Mgrid 应用出自 SPECOMP2001 测试集,均已通过 Brook+语言移植到 GPU 平台.Swim 应用实现二维浅水波方程求解器,该程序由 3 个核心计算过程和一个并行规约过程组成,本文将 3 个核心计算过程映射在 GPU 执行,规约过程由 CPU 完成.Mgrid 应用采取多重网格方法实现三维 Possion 方程求解器,该程序由 4 个核心计算过程组成.本文将计算量最大的程序段映射在 GPU 上完成,而将小规模程序段映射在 CPU 上完成.

Table 2 Experimental applications

表 2 测试用例介绍

Benchmark [abbr.]	Description	Language	Problem size
BinomialOption [BO]	Binomial option pricing model	OpenCL	262 144
BlackScholes [BS]	Black-Scholes model for European options		1048576
DCT	Discrete cosine transform		4096×4096
MatrixMultiplication [MM]	Matrix multiplication		4096×4096
NBody [NB]	Particles simulation		40 960
MonteCarloAsian [MCA]	Monte Carlo analysis		4 096
SWIM	Shallow water equation solver	Fortran & Brook+	2048×2048
Mgrid [MG]	3D Possion equation solver		256×256×256

4.3 实验评测

图 4 给出了 CPU 与 GPU 在执行时间和能耗开销上的对比图,其中,所有结果都是以 CPU 为基准的归一化值.从图中可以看出,所有应用程序在 GPU 上都可以获得较好的性能提升,尤其是计算密集性较高的应用 BO,BS,DCT 和 MM.与性能提升相比,GPU 在能耗上的优化相对较小,约为前者的 53%.可以看出,GPU 在具有高性能的同时,其功耗开销也相对较大.同时,由于微体系结构上存在的差异,CPU 和 GPU 在不同应用程序中体现出不同的能效特征,因此,应根据处理器在具体应用中的实际能效优化系统整体功耗.

© 中国科学院软件研究所 http://www.jos.org.cn

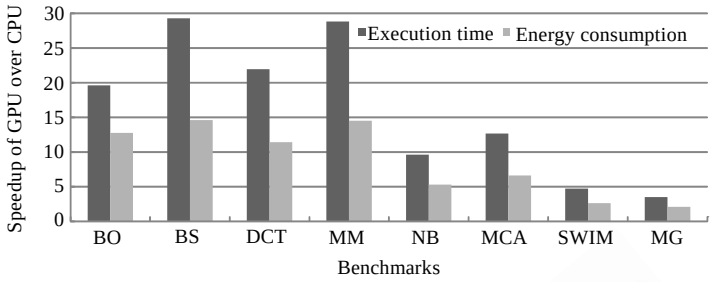


Fig.4 Execution time and energy consumption comparison between CPU and GPU

图 4 CPU-GPU 执行时间与能耗开销对比图

基于 OpenCL 语言实现的 6 个应用程序都仅由一个 Kernel 程序组成,并且同时支持 CPU 和 GPU 平台.因此,我们以这 6 个应用程序检验并行段能耗优化效果.我们通过分析和实验测得程序在异构系统上的最优执行时间,记为 T.通过放松时间约束,评估在执行时间不超过 $T(1+\beta)$(β称为放松因子)的条件下系统最优能耗.图 5 给出了最优能耗随时间放松因子的变化规律.可以看出:随着放松因子的增大,系统能耗明显降低;最大能耗优化效果出现在β小于 20%以内,当β大于 30%后,能耗优化空间相对较小.通过分析我们发现,主要原因包含两个方面:(1) 随着处理器频率的降低,访存逐渐成为系统的性能瓶颈,因此,处理器难以通过降频的方式有效利用放松的时间约束;(2) 处理器可选的运行频率级别较少(如 GPU 仅包含 3 个可选频率),缩小了细粒度能耗优化空间.

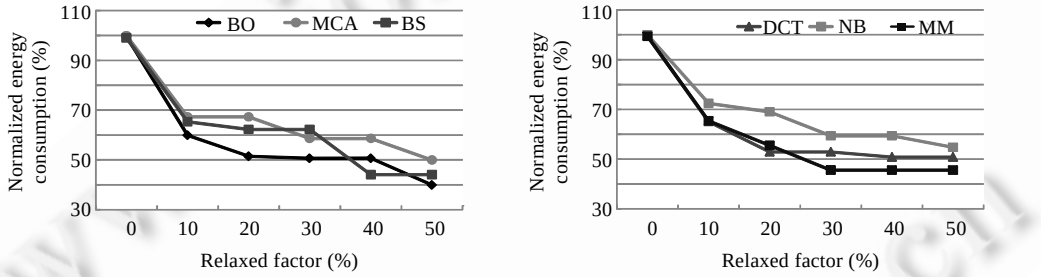
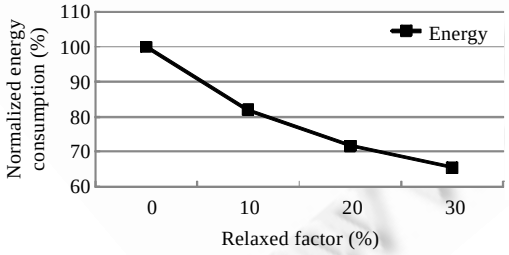


Fig.5 Energy consumption under different performance constraints for single Kernel

图 5 单 Kernel 程序能耗随约束时间的变化

图 6 和图 7 分别给出了 Mgrid 和 Swim 应用中最优能耗随约束时间的变化规律,图中右侧分别给出了在不同放松因子的约束下,所有程序段中处理器的运行频率.由于处理器频率存在最小值,因此,当处理器频率降低到最小值后增大放松因子不会带来能耗的节省,故图中仅列出了当处理器频率小于等于最小值时的情况.

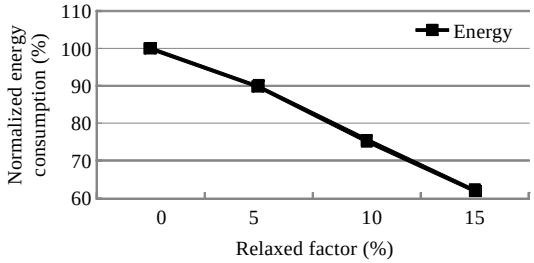


	GPU				CPU
Re.Factor	Resid	Rprj3	Interp	Psinv	Other
0%	0	0	0	0	0
10%	1	1	1	1	0
20%	2	1	1	2	0
30%	2	2	2	2	3

Fig.6 Energy consumption under different performance constraints for multi-kernel program (Mgrid)

图 6 多 Kernel 程序能耗随约束时间的变化(Mgrid)

© 中国科学院软件研究所 http://www.jos.org.cn



	GPU			CPU
Re.Factor	calc 1	calc 2	calc 3	Reduction
0%	0	0	0	0
5%	1	1	0	0
10%	2	1	1	1
15%	2	2	2	3

Fig.7 Energy consumption under different performance constraints for multi-kernel program (Swim)

图 7 多 Kernel 程序能耗随约束时间的变化(Swim)

在 Mgrid 应用中,CPU 完成计算量较小的计算过程,其执行时间和功耗开销都相对较小;而 GPU 在完成大粒度计算过程中功耗开销较大.从图 6 的表中可以看出,在给定约束时间的条件下,GPU 完成的 Kernel 计算过程的运行频率优先被降低.在 Swim 应用中,GPU 完成的 calc 1,calc 2 和 calc 3 计算过程都是访存密集型程序,尤其是 calc 3 程序;与之相应地,calc 3 程序的动态功耗最小,因此,与另外两个程序相比,该程序的降频操作排在最后.与单 Kernel 程序的优化效果类似,在处理器频率级数较少的情况下,处理器能耗优化空间相对较小.如图 6 和图 7 所示,当执行时间分别放松 30%和 15%后,各程序段中处理器频率已达到最小值,无法通过 DVFS 方法降低动态功耗产生的能量开销.

下面我们通过一个案例来评估和分析临界区操作对系统能耗开销的影响.该测试用例通过 OpenMP 并行编程模型实现,如图 8 所示,其并行段由一个可完全并行段和一个临界区段组成.以该程序为基础,我们可以在不改变并行段问题规模的条件下,调节临界区段在并行段中的比例(临界区操作比例σ)来评估临界区操作对并行段能耗优化的影响.该实验平台为表 1 中给出的 Intel Core I7 920 Quad-Core CPU,编译器为 gfortran.

```
Initialize input array A
!$omp parallel
  do i=1, n
    //full parallel code
    update A
    //critical section
    !$omp critical
      update A
    !$omp end critical
  enddo
!$omp end parallel
```

Fig.8 An example application with critical section

图 8 临界区测试用例

首先,我们检验处于临界区等待状态的处理器功耗开销.为了避免其他操作引入的开销,我们使并行段完全由临界区操作构成(即σ=1),因此,任一时刻有且仅有一个处理器执行有效计算.通过修改环境变量(*OMP_NUM_THREADS*)可以调节并发执行的线程数量.表 3 给出了处理器动态功耗开销随线程数量的变化情况,可以看出:随着线程数量的增加,处于临界区等待状态的线程数量也随之增加;然而,处理器总动态功耗并未明显变化.可以看出,处于等待状态的处理器并未明显产生额外功耗开销.

Table 3 Dynamic power consumption with different thread numbers

表 3 动态功耗开销随线程数的变化情况

Number of threads	1	2	3	4
Power consumption (w)	18.7	18.2	18.3	18.2

定理 2 通过分析方法发现,在时间约束一定的条件下,处理器最优运行频率将随着临界区比例的增大而增大.我们通过调节该测试用例中临界区操作的比例,检验处理器频率的变化情况,如图 9 所示.该测试以处理器在

© 中国科学院软件研究所 http://www.jos.org.cn

无临界区操作(σ=0)且处于最低运行频率(1.6GHz)时的执行时间为时间约束.作为对比,图中给出了 3 组数值,其中,理论值表示定理 2 中确定的处理器频率;由于处理器只支持有限个离散的频率值,因此实际频率为大于理论频率的最小物理频率值,记为物理值;最优值的计算方式是根据离散频率值下的执行时间,通过频率与时间的关系获得的理想频率,该值可能为实数.由图 9 可知,随着临界区比例的增大,处理器最优频率将相应提高,而且本文建立的处理器选择方法可以有效适应临界区的变化,如图中两曲线所示.当临界区比例达到 50%时,理论频率已经超过该处理器支持的最高频率,故此时物理值未列出.

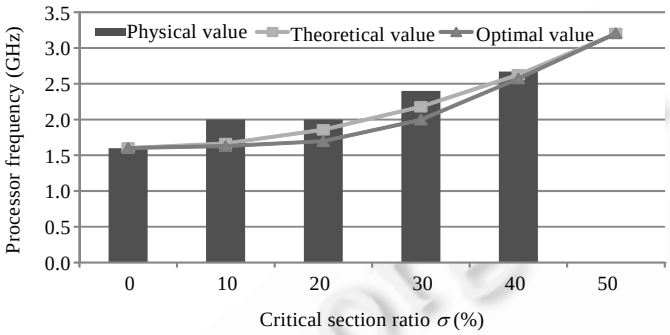


Fig.9 Operating frequency with different critical ratios under a fixed performance constraint

图 9 时间约束下处理器运行频率随临界区比例的变化情况

通过定理 2 的分析发现,在执行时间一定的条件下,使用细粒度锁降低临界区冲突概率,可以降低处理器运行频率,进而节省系统能耗开销.因此,我们将图 8 示例程序中的临界区拆分为两个临界区,并分别指定不同的临界区名称,因此,拆分后的临界区在不同的处理器上可以并发执行.此时,临界区冲突概率降低为 50%.图 10 给出了不同临界区比例下的能耗优化效果,所有结果都是相对原能耗值的归一化结果.图中柱状图上方给出了不同临界区比例下的处理器运行频率.对比图 9 中的频率值可以看出,降低临界区冲突概率为降低处理器频率提供了优化空间,进而降低了系统能耗开销,如图 10 所示.

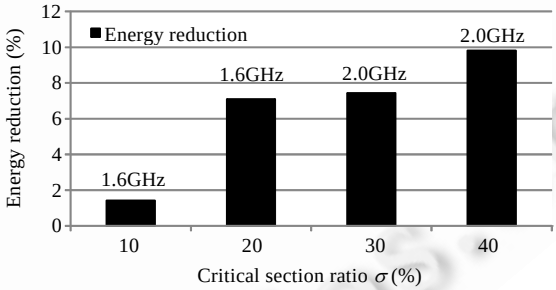


Fig.10 Energy reduction with fine-grained locking

图 10 通过细粒度锁降低系统能耗开销

5 相关工作

文献[3]基于 Amdahl 定律建立了传统多核、同构众核和异构众核体系结构的性能模型,通过对比分析,作者认为,集成有一个高性能计算核心和大量结构简单的高能效计算核心的异构体系结构,可以在保证串行执行效率的同时,有效提高并行处理能力.但是,文中并未涉及异构多处理器的能耗问题.随着多核处理器的发展,功耗问题逐渐成为制约处理器性能提升的主要瓶颈.文献[4]在文献[3]的基础上分析了 3 种多核体系结构的效能差异,包括同构多核、同构众核和异构众核体系结构,验证了异构体系结构较之同构体系结构具有更高的能效优

© 中国科学院软件研究所 http://www.jos.org.cn

势.但是,文献[3,4]都是在假设处理器具有固定的速度和功耗的基础上展开研究,并未考虑处理器动态调频等优化技术的影响.文献[5]面向同构多核体系结构建立了程序性能与能耗的关系,特别地,分析了在给定加速比约束的情况下如何调节处理器频率,以达到能耗最优;同时,作者分析了核心动态关闭技术对功耗优化的效果.文献[7]在文献[3]的基础上考虑了并行段中临界区对性能的影响,作者指出,在考虑临界区操作的影响下,异构体系结构加速比低于文献[3]中给出理想结果;同时指出,异构体系结构中应使用性能适中的并行处理核心,以提高临界区的执行效率.本文在上述研究的基础上,分析了异构体系结构中程序性能与能耗的关系,给出了时间约束下异构并行体系结构达到能耗最优的条件,并给出了相应的调度算法.基于文献[7]提出的性能模型,本文分析了临界区操作对异构并行体系结构能耗优化的影响.

利用处理器空闲周期,通过动态电压调节技术降低芯片功耗,是软件低功耗优化的重要途径之一.随着功耗问题的日趋严峻,其应用范围已经逐渐从实时应用领域扩大到通用计算,乃至高性能计算领域.文献[8]提出一种基于反馈驱动的多线程管理机制,针对同步受限或访存带宽受限应用程序,通过关闭冗余处理器核心,在不影响性能的情况下有效降低能耗开销.实验表明,该方法可以较为准确地判定同步受限或访存带宽受限程序,并选择合理线程数量.文献[9]首次提出了结合动态电压调节和动态核心关闭技术的二维低功耗优化方法,这是一种基于启发式的搜索策略,在核心数量和频率空间中搜索最优配置.设处理器具有 N 个处理器核心并支持 L 个运行频率,则穷举算法的复杂度为 LN,而作者给出的启发式搜索算法复杂度为 $a\lg N$(其中,$a<L$).文献[10]提出了基于硬件计数器的程序性能预测模型,在模型的指导下求解性能最优的处理器核心数量及其运行频率,在提高程序性能的同时降低芯片功耗.上述 3 种方法都是基于搜索或模型的动态方法,本文是通过理论分析方法研究异构并行系统的能耗优化问题.

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[11]提出了一种动态的异构系统任务划分方法,通过动态采样程序在不同计算单元上的执行时间,根据执行速度划分并行计算任务,以达到全系统性能最优.实验结果表明,该方法可以有效提高异构协同并行计算能力,更为高效地发挥异构系统的性能优势.对于大量不规则计算应用,文献[12]提出一种多GPU自适应负载平衡手段,通过在CPU和GPU间建立任务队列模型,GPU可以根据本地忙闲状态自适应地选择任务执行.文献[13]提出一种结合任务划分和任务窃取的负载平衡策略,在综合考虑任务亲和性和处理器差异性的基础上,指导CPU-GPU间任务调度.目前,基于CPU-GPU异构系统的研究主要围绕性能优化方面,异构并行系统的能耗优化研究尚处于起步结点,本文正是针对该问题展开研究.

6 结束语

面向异构并行系统的功耗优化问题,是高效开发异构系统能效优势的关键问题之一.本文针对由串行段和并行段组成的一般程序模型,建立了异构并行系统能耗优化模型.针对并行段程序,分析得出能耗最优时处理器间满足的关系,并基于该性质给出了能耗最优的处理器频率选择算法.在保证程序段内能耗最优的基础上,本文将时间约束下的全程序能耗最优问题归纳为一般多元极值问题.为降低计算复杂度,分析了同构程序段的能耗优化问题,给出了时间约束下同构程序段达到能耗最优的条件,并基于此给出了同构程序段中处理器的频率选择算法.最后,本文以CPU-GPU异构并行系统为平台,验证了该优化模型的有效性.

References:

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

[2] http://www.top500.org/lists/2010/11

[3] Hill MD, Marty MR. Amdahl's law in the multicore era. Computer, 2008,41(7):33−38. [doi: 10.1109/MC.2008.209]

[4] Woo DH, Lee HHS. Extending Amdahl's law for energy-efficient computing in the many-core era. Computer, 2008,41(12):24−31. [doi: 10.1109/MC.2008.494]

[5] Cho SY, Melhem RG. On the interplay of parallelization, program performance, and energy consumption. IEEE Trans. on Parallel and Distributed Systems, 2010,21(5):342−353. [doi: 10.1109/TPDS.2009.41]

© 中国科学院软件研究所 http://www.jos.org.cn

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

[7] Eyerman S, Eeckhout L. Modeling critical sections in Amdahl's law and its implications for multicore design. In: Proc. of the 37th Annual Int'l Symp. on Computer Architecture (ISCA 2010). New York: ACM Press, 2010. 362−370. [doi: 10.1145/1815961.1816011]

[8] Suleman MA, Qureshi MK, Patt YN. Feedback-Driven threading: Power-Efficient and high-performance execution of multi-threaded workloads on CMPs. In: Proc. of the 13th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS XIII). New York: ACM Press, 2008. 277−286. [doi: 10.1145/1353534.1346317]

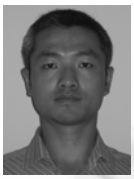
[9] Li J, Martinez JF. Dynamic power-performance adaptation of parallel computation on chip multiprocessors. In: Proc. of the 20th Int'l Symp. on High-Performance Computer Architecture (HPCA 2006). 2006. 77−87. [doi: 10.1109/HPCA.2006.1598114]

[10] Curtis-Maury M, Shah A, Blagojevic F, Nikolopoulos DS, de Supinski BR, Schulz M. Prediction models for multi-dimensional power-performance optimization on many cores. In: Proc. of the 17th Int'l Conf. on Parallel Architectures and Compilation Techniques (PACT 2008). New York: ACM Press, 2008. 250−259. [doi: 10.1145/1454115.1454151]

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

[12] Chen L, Villa O, Krishnamoorthy S, Gao GR. Dynamic load balancing on single- and multi-GPU systems. In: Proc. of the 2010 IEEE Int'l Symp. on Parallel & Distributed Processing (IPDPS). 2010. 1−12. [doi: 10.1109/IPDPS.2010.5470413]

[13] Hermann E, Raffin B, Faure F, Gautier T, Allard J. Multi-GPU and multi-CPU parallelization for interactive physics simulations. In: Proc. of the 16th Int'l Euro-Par Conf. on Parallel Processing: Part II (Euro-Par 2010). Berlin, Heidelberg: Springer-Verlag, 2010. 235−246.



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

唐滔(1984－),男,博士生,主要研究领域为计算机体系结构,编译技术.

杨学军(1963－),男,博士,教授,博士生导师,主要研究领域为并行计算机体系结构与编译,容错并行算法,流处理器体系结构与编译.

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.

© 中国科学院软件研究所 http://www.jos.org.cn