

功耗受限情况下多核处理器能效优化方案

邱晓杰, 安 虹, 陈俊仕, 迟孟贤, 金 旭

(中国科学技术大学 计算机科学与技术学院, 合肥 230027)

摘 要: 将处理器功耗控制在预算以下有助于降低散热成本和提升系统稳定性, 但现有功耗优化方案大多依赖线下分析得到的先验知识, 影响实用性, 而集中式搜索最优策略的算法也存在复杂度过高的问题。为此, 提出功耗优化方案 PPCM。利用动态电压频率调整(DVFS)技术控制 CPU 功耗在预算内以提高处理器能效。同时, 将功耗控制和功耗分配解耦合以提高灵活性。采用动态调整的线性模型估计功耗, 通过反馈控制技术对其进行调节。以计算访存比为指标在应用间分配功耗, 并考虑多线程应用特征进行线程间功耗分配。实验结果表明, PPCM 比 Priority 算法速度平均提高 10.7%, 能耗平均降低 5.1%, 能量-延迟积平均降低 14.3%。与 PCMCA 算法相比, 其速度平均提高 4.5%, 能量-延迟积平均降低 5.0%。

关键词: 功耗控制; 功耗分配; 能效优化; 动态电压频率调整; 计算访存比; 线程关键度

中文引用格式: 邱晓杰, 安 虹, 陈俊仕, 等. 功耗受限情况下多核处理器能效优化方案[J]. 计算机工程, 2017, 43(4): 39-45.

英文引用格式: Qiu Xiaojie, An Hong, Chen Junshi, et al. Energy Efficiency Optimization Scheme for Multi-core Processor Under Power Consumption Constraint[J]. Computer Engineering, 2017, 43(4): 39-45.

Energy Efficiency Optimization Scheme for Multi-core Processor Under Power Consumption Constraint

QIU Xiaojie, AN Hong, CHEN Junshi, CHI Mengxian, JIN Xu

(School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)

[Abstract] Keeping processor power consumption under budget can reduce the cost of cooling and improve the system reliability. Most existing energy efficiency optimization schemes are profile-based offline schemes, which may reduce practicality. Furthermore, centralized algorithms which exhaustively search for the optimal configuration may be too complex. In this paper, a power consumption optimization scheme PPCM is proposed, which utilizes Dynamic Voltage and Frequency Scaling(DVFS) to control CPU power consumption under budget and improve energy efficiency. PPCM decouples power consumption control and power consumption allocation to improve flexibility. It uses a dynamic linear model to estimate power consumption and manages it based on feedback control technology. It uses the ratio of computation to memory access as an indicator to allocate power consumption among applications, and then considers the features of multithread application and allocates power consumption among threads. Experimental results show that PPCM outperforms Priority algorithm by 10.7% in speed in average, 5.1% in energy saving in average and Energy-Delay Product(EDP) is reduced by 14.3% in average. It is superior to PCMCA by 4.5% in speed in average and 5.0% in EDP in average.

[Key words] power consumption control; power consumption allocation; energy efficiency optimization; Dynamic Voltage and Frequency Scaling(DVFS); ratio of computation to memory access; thread criticality

DOI: 10.3969/j.issn.1000-3428.2017.04.007

基金项目: 国家“863”计划项目(2012AA010901)。

作者简介: 邱晓杰(1990—), 男, 硕士研究生, 主研方向为处理器功耗优化技术; 安 虹, 教授、博士生导师; 陈俊仕, 博士研究生; 迟孟贤、金 旭, 硕士研究生。

收稿日期: 2016-04-15 **修回日期:** 2016-05-25 **E-mail:** qxj@mail.ustc.edu.cn

0 概述

多核处理器设计和运行面临功耗和温度的挑战。多核处理器上同时执行的任务复杂多样,功耗快速变化,很有可能超出处理器的功耗预算,从而超过其散热能力导致过热。温度涉及到很多处理器内部的机制,过热可能导致暂时或者永久性失效,如时序错误、处理器寿命减少,甚至更严重的芯片失效。多核处理器热量空间分布不一致也会降低散热系统的效率,更严重的是,机械应力(mechanical-stress)的存在可能损害芯片的稳定性。另外,处理器在22 nm以后将会遇到 Dark Silicon 现象^[1],即芯片上大量晶体管将存在功耗问题而无法被供电。因此,通过功耗管理将有限的功耗分配给合适的部件变得尤为重要。现代处理器提供的多种功耗调节机制可以实现运行时管理。动态电压频率调整(Dynamic Voltage and Frequency Scaling, DVFS)是最有效的方法之一,它能降低处理器电压和频率,从而降低功耗和温度。设计基于 DVFS 的功耗管理算法面临以下问题:1)核数增加和频率调节范围的扩大,使得兼顾功耗控制和能效优化的集中式算法复杂度过高;2)功耗和平台与应用特征密切相关,在已有的工作中,大量的算法模型依赖离线训练,这限制了算法的使用场景。由于训练负载和实际运行负载不同,离线分析会降低算法的适应能力,因此如何设计通用可靠、可移植性高的功耗控制和优化策略至关重要。

针对上述问题,本文提出 PPCM 方案,针对多应用多线程负载场景,基于 DVFS 技术进行功耗控制和能效优化:1)将功耗控制和功耗分配解耦合,使两者独立地根据各自目标进行优化,降低复杂度,增加灵活性。2)基于频率和功耗的关系模型,通过反馈方式指导 DVFS 调节,实现功耗控制,并可通过在线调整模型参数,提高该方法在不同平台和面对不同特征应用时的适应能力。3)针对多应用多线程环境,分层进行功耗分配。同时以计算访存比为功耗分配指标,兼顾公平性,提高系统能效。

1 相关研究

Intel Montecito 双核处理器配备了一个控制系统,包含功耗、温度传感器以及专用的微控制器,可以通过芯片级 DVFS 技术实现功耗和温度控制^[2]。之后,更多的商业处理器提供了每个核独立调节频率的能力,使得功耗、温度控制更加精细。启发式算法被很多功耗控制算法使用^[3-6]。启发式方法包括通过试错(trial-and-error)方式调节处理器频率^[3,6],以及通过功耗-性能模型穷举每个核合适的配置,找到优化方案。后续的工作揭示了启发式算法的不足,包括算法的开销大、控制不精确以及性能受限^[7-8]的问题。为

增加对模型出错的鲁棒性,以及为算法性能提供理论保证,研究者提出了一些基于反馈控制理论的芯片功耗控制方法。文献[9]针对数据库服务器系统,使用闭环反馈控制进行功耗封顶。文献[10-11]提出的功耗控制策略,先将功耗分配给各个独立电压调节单元以最大化 Chip 性能,然后本地调节单元使用比例-积分-微分控制方法(PID 方法)调节本地频率以满足功耗预算。由于该方法在分配功耗时没有考虑到各个调节单元能达到的功耗范围,因此可能会给调节单元分配无法达到的功耗预算,从而降低功耗控制和性能优化的准确度。文献[12]提出了一个基于模型预测控制(Model Predictive Control, MPC)的集中式控制方法,其中处理器核频率和芯片功耗的关系被抽象成线性的无记忆的动态系统,输入参数可在运行时获得。然而,由于该方法是集中式的并且需要做矩阵运算,导致其计算量和运行开销较大。文献[8]提出了一种多核平台上针对单线程、多线程负载的功耗控制方法。该方法使用整体控制方式,通过调整聚合频率(所有核的频率总和)来保证功耗不超过限制功耗。通过将聚合频率分配给不同的核来最大化系统性能。但聚合频率的分配没有考虑核所能调节的频率范围,计算出的频率可能会是无法调节到的频率,从而影响功耗控制和性能表现。而且该方法在设计控制器和估测负载性能表现(为了最大化公平调度指标)时都需要预先执行应用以获得相关数据,这在实际场景中比较受限。

除了单独利用 DVFS 进行功耗控制之外,一些研究考虑了内存、Cache 等部件的功耗^[13-14],通过在多个部件之间迁移功耗使得系统更为高效。还有一些研究协同考虑 DVFS 技术和其他管理技术,如线程迁移、线程聚合等^[15-16]。文献[17]针对 CPU-GPU 异构系统进行功耗控制,通过建立异构系统能效评估方法实现 CPU 和 GPU 之间的功耗分配。

2 功耗受限情况下的能效优化策略

在计算机系统整体功耗中,处理器功耗占据主导地位,占比超过 1/3^[18]。而且,CPU 功耗变化幅度很大,从数瓦特到数十瓦特,是引起系统功耗波动的重要原因。因此,本文重点关注 CPU 的功耗控制及优化研究,希望在不超功耗预算的情况下优化系统能效。这里的功耗预算软性要求,允许在短时间内少量超过预算。

本文的能效优化策略分为 2 个部分:1)全局功耗控制,保证全局功耗不超过功耗预算且接近预算功耗;2)功耗分配,根据能效优化目标,采用合适的策略,将功耗分配给各个独立的调节单元。

2.1 全局功耗控制

2.1.1 处理器功耗模型

处理器功耗主要由动态功耗和静态功耗两部

分组成,可用 $P = P_s + P_d$ 表示。静态功耗由漏电流造成, $P_s = I_{leak} \times V_d$ 。处理器动态功耗可近似使用公式 $P_d = CAV^2F$ 表示,其中, C 为芯片电容; A 为芯片上活跃的门的比例; V 为供给电压; F 为时钟频率。

一般地,越高的时钟频率要求的供给电压越高,频率与电压成正比关系,因此,降低频率可以获得电压立方级收益。在实际处理器中,电压的变化幅度较小,而且由于静态功耗的存在,功耗和频率不是完美的立方级关系。本文收集了 SPEC CPU2006 程序集中所有程序在不同频率下执行的平均功耗,图1列举了5个应用程序的功耗和频率关系,可以发现,功耗和频率近似呈线性关系,其他未列举的应用也都表现出这种特征。

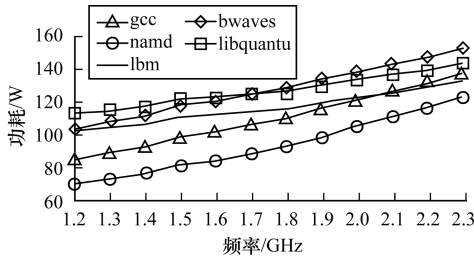


图1 程序在不同频率下的平均功耗

文献[19]也展示了处理器功耗和频率近似的线性关系。本文使用 $P = a \times F + b$ 来近似估计处理器功耗,其中, P 为处理器功耗; F 为处理器频率总和。同时还可以发现:1)在相同频率下,不同应用的功耗不同,并且相差较大;2)不同应用下功耗随频率变化而变化的速度也不相同。这说明使用线下模型估计功耗时,不同应用 a, b 值是不同的。当系统运行多应用时,处理器功耗与频率关系就是各应用特征叠加之后的效果,也就是说在公式 $P = a \times f + b$ 中, a 和 b 的值随着平台以及应用特征发生改变。

2.1.2 基于反馈的功耗调节机制

本文采用上节所述的 $p = a \times f + b$ 模型来估计处理器功耗,使用反馈控制技术来调节功耗,在每个采样周期结束时,根据实测功耗和预算功耗的差值,得到出下个周期的调节策略。使用 f_{min}, f_{max} 分别表示核能调节的最低频率和最高频率, f_i 为核 i 的频率, N 为处理器核数, Pb 表示功耗预算, $Pc(k)$ 表示周期 k 实际的功耗, $F(k)$ 表示周期 k 用于分配的频率总和, $F(k) = \sum_{i=1}^N (f_i - f_{min})$ 。在 k 周期末,功耗差值 $Pd(k) = Pb - Pc(k)$,则下个周期可用于分配的频率总和 $F(k+1) = F(k) + Pd(k)/a$ 。 a 值与系统平台和应用特点有关,可以通过线下训练获得。然而,线下训练得到的 a 与训练样本有关,并不能很好地适应复杂多变的运行时环境,而且不同系统平台的特

点不一样, a 值也不相同,线下训练会影响该方法跨的平台使用效果。

本文设计可动态调整模型参数的方法。首先在初始时给 a 设定一个初值,在一个合理的范围内即可。然后在程序运行过程中动态地调整 a 。具体如下:当 Pd/Pb 大于阈值 δ 时,调整 $a = 0.5 \times a + 0.5 \times a'$ 。其中, $a' = (Pc - b)/F$; b 为处理器在最低频率时的功耗。不直接采用 a' 取代 a 是为了防止偶然的波动,保持平滑。阈值 δ 的设定可根据实际需求,如对功耗抖动幅度的忍受程度,本文设定 $\delta = 0.1$ 。在最低频率工作时的功耗 b 比较容易获得,在初始阶段时,处理器在最低频率上运行一段时间,采集处理器的平均功耗。

图2描述了调整参数及确定可用分配频率 F 的流程。采用动态调整模型参数,一方面,当移植到不同机器上时,不需要离线分析去获得精确的 a 值,只需设置一个初值,在运行过程中, a 值就会趋向于合理的值;另一方面,应用在执行过程中会有阶段性,动态调整 a 可以使算法适应不同的程序阶段。

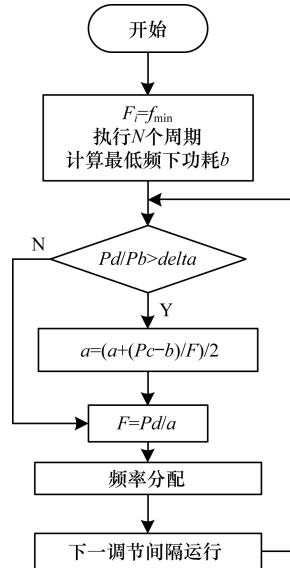


图2 参数调整及确定可用分配频率的流程

2.2 应用间功耗分配

在确定处理器可用分配频率总和 F 之后,需要将频率分配给各个独立的调节域,即可实际调节频率的单元。在本文中,对频率的分配实际上就是对功耗的分配,下文中不严格区分两者。不同的处理器支持的可调节频率的粒度不同,早前的处理器只支持 Chip 级频率调节,如今高端的处理器已经支持每个处理器核独立调节频率。本文提出的功耗分配策略针对每个核可独立调节频率的平台,并且考虑多线程应用执行特点。多线程应用在多个处理器核上同时执行时,线程之间可能存在交互。以前大量

工作将每个核独立看待,不利于多线程应用的执行。本文使用分层分配频率的方法,先将应用作为单位进行频率分配,然后在应用内部以处理器核作为单位进行频率分配。

在应用间功耗分配中,将同一个应用占据的处理器核看作一个组,认为是一个可调节频率的调节单元,其可调节的频率范围为 $0 \sim N_i \times (f_{\max} - f_{\min})$, N_i 为第 i 个应用占用的核数。对于没有程序执行的处理器核,将其始终设为最低频率。通过将功耗分配给加速效果更明显的应用,从而缩短执行时间,提高能效。

文献[20]认为,指令执行时间可以分成两部分:计算时间和阻塞时间。阻塞时间是指由于访存延迟造成的停顿时间,该段时间不随处理器频率变化而变化。其他的时间称为计算时间,计算时间随着处理器频率变化而变化。在实际应用中可以近似认为阻塞时间等于访存开销时间。使用 T_{comp} 表示计算时间, T_{mem} 表示访存时间,则指令执行时间表示为:

$$T_{\text{total}} = T_{\text{comp}} + T_{\text{mem}} \quad (1)$$

一般地,可以认为计算时间与处理器频率成反比,即有:

$$\frac{T_{\text{comp}}(f')}{T_{\text{comp}}(f)} = \frac{f}{f'} \quad (2)$$

因此,可由当前频率 f' 预测在频率 f 下的指令执行时间 $T_{\text{total}}(f)$,如式(3)所示。

$$T_{\text{total}}(f) = T_{\text{mem}}(f') + T_{\text{comp}}(f') \times \frac{f'}{f} \quad (3)$$

$$\frac{T_{\text{total}}(f)}{T_{\text{total}}(f')} = 1 + \frac{f'}{f} \times \frac{T_{\text{comp}}}{T_{\text{total}}} - \frac{T_{\text{mem}}}{T_{\text{total}}} \quad (4)$$

从式(4)可以看出,当计算时间所占比例越大,通过调高频率获得的加速比越大,或者说计算密集型程序随着频率升高性能收益越高。基于此,希望将有限的功耗尽量分配给计算密集型应用。如果贪心地将功耗都分配给计算密集型应用,则访存密集的应用可能一直处于最低速的运行状态,严重影响公平性。因此,功耗按照计算密集程度等比例地分配给各应用。用 r_i 表示第 i 个应用的计算访存比, $r_i = T_{\text{comp}}/T_{\text{mem}}$,要根据 r_i 等比例分配频率,理想的 f_i , r_i 需要满足以下 3 个条件:

$$0 < f_i < f_{\max} - f_{\min} \quad (5)$$

$$F = \sum_{i=1}^N f_i \quad (6)$$

$$f_i = r_i / \sum_{i=1}^N r_i \times F \quad (7)$$

实际上,受到频率调节范围的约束,上述公式无法全都成立,只能尽量按照 r_i 之间的比例分配频率。采用先排序再按比例分配的方法,先根据 r 排序,从 r

最大的应用开始,按其在 $R(R = \sum_{i=1}^N r_i)$ 中所占比例分配频率,如果其分得的频率超过可以使用的最大频率,则取最大频率;然后选择 r 次大的应用,在剩余频率中按剩余占比分配。具体如算法 1 所示。

算法 1 频率映射算法

输入 每个调节单元的分配系数 $r[N]$,总频率 F

输出 每个调节单元分得频率 f_i

1. Sort (r)//descending order
2. $R = \text{sum}(r)$
3. Repeat://until selected all r_i from r
4. Select r_i from r ;
5. $f_i = r_i/R$
6. if $f_i > f_{\max}$: $f_i = f_{\max}$
7. $R = R - r_i$

通过该算法,可使可用分配频率 F 全部被使用,并且每个应用分得的频率比例接近 r 之间的比例。一般地,可以将 r 称为分配系数,当采用其他的指标给 r 赋值之后,仍可使用该算法进行分配。

2.3 组内功耗分配

组内频率分配是进一步将频率分配给每个执行相同应用线程的处理器核。多线程在执行时,由于调度竞争、线程间通信等,导致执行进度不一,从而影响应用完成时间。加速多线程应用关键在于找到关键线程并对其进行加速。文献[21]给出了设计基于线程访存开销的关键度评价方法,其设计思想为:认为访存开销大的线程受到访存影响更严重,从而执行进度较慢。对执行进度慢的线程进行加速,有助于减少其他线程的等待时间,缩短整个程序的运行时间,降低功耗。该方法适用于有同步路障和无同步路障的情况,借助当今处理器普遍配备的性能监测单元(PMU),能够方便地在商用机器上实现。本文采用该方法作为应用内部线程间功耗分配的基础。应用 i 线程 j 的关键度由式(8)计算。

$$Cr_{ij} = N(L1_{\text{miss}}) + \frac{(L1L2_{\text{penalty}}) \times N(L1L2_{\text{miss}})}{L1_{\text{penalty}}} \quad (8)$$

其中, $N(L1_{\text{miss}})$, $L1_{\text{penalty}}$ 分别为 L2 Cache 上的命中次数和命中开销; $L1L2_{\text{penalty}}$, $N(L1L2_{\text{miss}})$ 分别为 L2 Cache 上的缺失次数和开销。

如果直接使用式(8)中的关键度作为分配功耗的依据,可能导致线程间不公平。线程在一段时间内的访存流量与频率有关,一般地,频率越高,单位时间内访存越多。2 个行为相同的线程,某个时刻一个线程被赋予了更高的频率,则在之后的执行中,该线程的访存开销更大,从而被赋予更高的频率,导致 2 个线程的频率差距越来越大。不同频率下的线程直接按照访存开销分配功耗,有失公平。

所以,需要将访存开销统一到相同的频率上然后进行功耗分配。令 N_{data} 表示 T_{total} 时间段内总的访存数目, R 表示单位时间内访存流量, R 计算如式(9)所示。

$$R = \frac{N_{\text{data}}}{T_{\text{total}}} \quad (9)$$

则在频率 f' 下单位访存流量 $R_{f'}$ 为:

$$R_{f'} = \frac{N_{\text{data}}(f)}{T_{\text{comp}}(f) \frac{f}{f'} + T_{\text{mem}}(f)} \quad (10)$$

因此,在频率 f 下计算频率 f' 下的关键度 $Cr_{ij}(f')$ 为:

$$Cr_{ij}(f') = \frac{1}{1 - m + m \times \frac{f}{f'}} \times Cr_{ij}(f) \quad (11)$$

其中, m 为计算时间占总时间的比例。将所有线程的访存开销都统一到最低频率之后进行功耗分配。类似于上节应用间分配时遇到的问题,需要把范围更广的关键度映射为范围较小的频率,仍使用算法 1 进行频率分配。

3 实验评估

3.1 实验环境

本文使用的软硬件平台详细信息如表 1 所示。测试程序是从 SPEC CPU2006 和 PARSEC3.0 基准测试集中抽出部分程序进行组合,模拟真实系统的负载情况,包括单线程应用和多线程应用。表 2 为测试程序情况,数字代表该应用占用的核数(单线程应用则代表该应用的个数,多线程则代表该应用线程数目)。功耗控制程序以监控者的身份运行在后台,每个周期通过 PMU 单元读取硬件事件,包括指令发射数、时钟周期数、各级缓存命中及缺失情况,在每个周期末使用上述方法进行能效优化。能效采用能量-延迟积(Energy-Delay Product, EDP)为衡量指标, $EDP = \text{Time} \times \text{Energy}$, 其中, Time 为应用完成时间, Energy 为应用完成所需能量。

表 1 实验环境

项目	配置
操作系统	Centos 7.0, Linux Kernel 3.10.0
CPU 型号	Intel(R) Xeon(R) E5-2650 v3
CPU 个数	2
每个 CPU 核数	10
CPU 支持的频率调节	1.2 GHz ~ 2.3 GHz, 每 0.1 GHz 为 1 个调节间隔
内存	64 GB
L1/L2/L3 Cache	32 KB/256 KB/25 600 KB × 2

表 2 实验测试集

组合	应用	特征
Mix1	10-gcc + 10-dealII	功耗波动大
Mix2	5-(Perlbenc + mcf + sople + sjeng)	计算密集 + 访存密集
Mix3	10-Streamcluster + 10-bodytrack	高同步路障 + 低同步路障
Mix4	4 (freqmine + ferret + streamcluster + volrend) + perlbenc + gcc + mcf + omnetpp	各类应用随机组合

本文选择文献[8]算法和 Priority 算法^[19]2 种策略进行对比。本文称文献[8]算法为 PCMCA, 该算法同样使用了分层分配功耗的思想, 但是较多地使用了离线训练, 是目前比较优秀的功耗控制及优化策略。Priority 算法实用性较高: 对每个核赋予一个优先级, 当功耗低于预算时, 选择最高优先级的核升频, 如果已经达到最高频, 则选择次高优先级的核, 以此类推; 如果功耗高于预算, 则选择最低优先级的核降频, 如果已经是最低频, 则选择次低优先级的核降频, 以此类推。Priority 算法将每个线程独立看待, 没有考虑到多线程应用的特点, 不能很好地适地多线程应用场景。

3.2 实验结果

3.2.1 功耗控制

在本文实验中, 对 4 个组合分别进行了测试, 并分析其功耗控制的精度以及波动程度。初始时, 将所有核设置在最低频率, 防止功耗超过预算, 控制周期设置为 50 ms。图 3 是 Mix1 执行时处理器功耗情况。设定功耗预算为 120 W, 其中虚线是固定处理器频率为 1.9 GHz 时处理器功耗情况, 可以发现即使在固定频率的情况下, 该组合的功耗变化也非常剧烈。实线是使用 PPCM 功耗控制策略后的结果, 可以看出其达到了稳定的功耗控制效果。

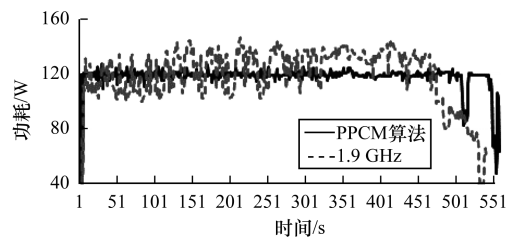


图 3 不同策略下 Mix1 的功耗

图 4 是 Mix2 分别在 PPCM 和 Priority 算法下的功耗情况, 设定功耗预算为 120 W。可以发现, Priority 算法由于一次调节一步, 因此当程序功耗起伏大时来不及调节, 从而导致较大的功耗波动。需要说明的是, 在运行后期, 由于某些应用已经执行完, 导致整体功耗下降, 即使所有线程所在核都调成最高频率也无法达到功耗预算, 这种情况本身也符合不超过功耗预算的要求, 但是会给评价 PPCM 功耗控制能力带来影响, 因此量化功耗控制能力时忽

略这段时间,但在评价系统性能时会考虑整体的执行时间。

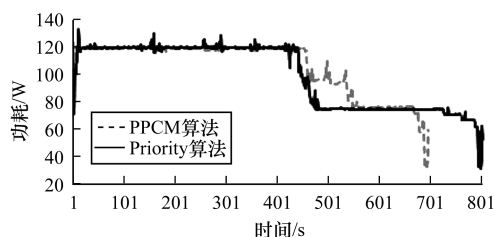


图4 不同策略下 Mix2 的功耗

表3是所有组合的功耗控制下的情况。可以看出,PPCM具有良好的功耗控制精度和稳定程度,与预算功耗的平均误差为0.4%,最大误差为0.7%。

表3 PPCM 对不同测试集功耗控制情况 W

组合	功耗预算	实际功耗	标准差
Mix1	120	119.10	2.10
Mix2	120	119.20	0.58
Mix3	110	109.70	1.33
Mix4	110	109.56	0.56

3.2.2 动态调整模型参数的效果

本节比较不同 a 值对功耗控制的影响。在PPCM中,修改 a 的不同取值方式。选择3种不同 a 值情况进行分析:

1) 固定 a 为适合实验平台的值,本实验平台对应为0.568;

2) 固定 a 为偏离实验平台特征较大的值,表示在不同平台上可能出现的 a 值误差较大的情况,设为0.1;

3) 使用本文提出的动态调节方法, a 初值偏离更严重,设为0.01。

在图5中,实线是 a 值偏小的情况下功耗情况,由于 a 值太小,因此在频率调整时幅度过大,导致功耗大幅度上下波动。带三角虚线是在有动态调整参数功能的情况下的功耗情况,即使其初始值偏离合适值更严重,通过刚开始的多次调整之后达到合适的 a 值,与 a 设定为0.568时取得了同样稳定的功耗控制效果。

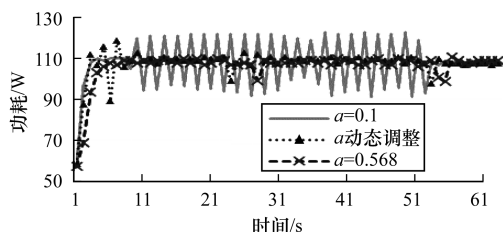


图5 a 不同时 Mix4 的功耗

3.2.3 系统性能

本节研究PPCM在能效方面的优化效果。如图6所示,将PPCM算法和PCMCA算法下程序的完成时

间分别归一化到Priority算法下进行比较。可以看出,Priority算法性能最差。PPCM相较于Priority算法,平均性能提升10.7%,最大提升18.1%。PPCM相较于PCMCA算法,平均性能提升4.5%,最大提升5.1%。

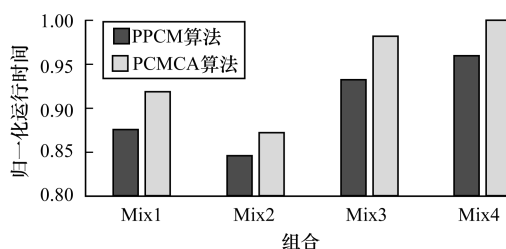


图6 不同策略的运行时间对比

图7是3种算法的归一化能耗对比情况。Priority算法的能耗仍然是最大的。相较于Priority算法,PPCM算法能耗平均降低5.1%,最大降低9.2%。PPCM与PCMCA的能耗基本相等,PPCM略优。

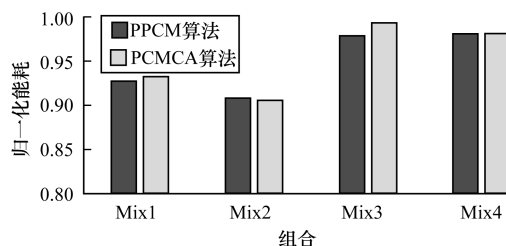


图7 不同策略的能耗对比

总体上看,PPCM是三者中性能最优的。与Priority算法相比,PPCM算法EDP平均降低14.3%。与PCMCA算法相比,PPCM在性能上优于PCMCA,而能耗近似相等,EDP平均降低5.0%。

4 结束语

处理器能效优化是工业界和学术界研究的重要领域,控制处理器功耗有助于提升系统稳定性,降低散热成本。本文提出面向多应用多线程的功耗优化方案PPCM。首先使用频率的线性模型估计处理器功耗,通过反馈机制将处理器功耗控制在预算以下;然后在应用间参考计算访存比进行功耗分配,在应用内部以访存开销为指标进行线程间功耗分配。PPCM重点考虑了应用特征变化带来的功耗模型误差,通过动态调整模型参数提高功耗调节的准确度。算法中所有模型参数都不需要离线分析,也不需要特定的机器信息,可移植性较强。实验结果表明,PPCM实际功耗与预算功耗平均误差为0.4%。与Priority算法相比,其速度平均提升10.7%,能耗平均下降5.1%,EDP平均降低14.3%;与PCMCA算法相比,速度平均提升4.5%,EDP平均降低5.0%。

下一步将考虑结合 CPU DVFS 和 CPU Hotplug 技术,通过选择最佳的 CPU 工作核数和 CPU 工作频率实现更好的能效优化,同时扩大 CPU 支持的功耗预算范围。

参考文献

- [1] Esmailzadeh H, Blem E, Amant R S, et al. Dark Silicon and the End of Multicore Scaling [C]//Proceedings of International Symposium on Computer Architecture. San Jose, USA: [s. n.], 2011:365-376.
- [2] McGowen R, Poirier C A, Bostak C, et al. Power and Temperature Control on a 90-nm Itanium Family Processor [J]. IEEE Journal of Solid-State Circuits, 2006, 41(1):229-237.
- [3] Isci C, Buyuktosunoglu A, Cher C Y, et al. An Analysis of Efficient Multi-core Global Power Management Policies: Maximizing Performance for a Given Power Budget [C]//Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture. Washington D. C., USA: IEEE Press, 2006:347-358.
- [4] Teodorescu R, Torrellas J. Variation-aware Application Scheduling and Power Management for Chip Multiprocessors [C]//Proceedings of the 35th Annual International Symposium on Computer Architecture. New York, USA: ACM Press, 2008:363-374.
- [5] Sartori J, Kumar R. Distributed Peak Power Management for Many-core Architectures [C]//Proceedings of DATA'09. Washington D. C., USA: IEEE Press, 2009:1556-1559.
- [6] Winter J A, Albonesi D H, Shoemaker C A. Scalable Thread Scheduling and Global Power Management for Heterogeneous Many-core Architectures [C]//Proceedings of the 19th International Conference on Parallel Architectures and Compilation Techniques. New York, USA: ACM Press, 2010:29-40.
- [7] Wang Yefu, Ma Kai, Wang Xiaorui. Temperature-constrained Power Control for Chip Multiprocessors with Online Model Estimation [C]//Proceedings of the 36th Annual International Symposium on Computer Architecture. New York, USA: ACM Press, 2009:314-324.
- [8] Ma Kai, Li Xue, Chen Ming, et al. Scalable Power Control for Many-core Architectures Running Multi-threaded Applications [C]//Proceedings of the 38th Annual International Symposium on Computer Architecture. New York, USA: ACM Press, 2011:449-460.
- [9] 杨良怀,阮忠孝,朱红燕,等.数据库服务器系统中一种有效的功率封顶机制 [J]. 计算机科学, 2015, 42(S2):490-496.
- [10] Mishra A K, Srikantaiah S, Kandemir M, et al. CPM in CMPs: Coordinated Power Management in Chip-multiprocessors [C]//Proceedings of International Conference on High Performance Computing, Networking, Storage and Analysis. Washington D. C., USA: IEEE Computer Society, 2010:1-12.
- [11] Mishra A K, Srikantaiah S, Kandemir M, et al. Coordinated Power Management of Voltage Islands in CMPs [C]//Proceedings of ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems. New York, USA: ACM Press, 2010:359-360.
- [12] Wang Xiaorui, Ma Kai, Wang Yefu. Adaptive Power Control with Online Model Estimation for Chip Multi-processors [J]. IEEE Transactions on Parallel and Distributed Systems, 2011, 22(10):1681-1696.
- [13] Ke Meng, Joseph R, Dick R P, et al. Multi-optimization Power Management for Chip Multiprocessors [C]//Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques. New York, USA: ACM Press, 2008:177-186.
- [14] Sharifi A, Mishra A K, Srikantaiah S, et al. PEPON: Performance-aware Hierarchical Power Budgeting for NoC Based Multicores [C]//Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques. New York, USA: ACM Press, 2012:65-74.
- [15] Cochran R, Hankendi C, Coskun A K, et al. Pack & Cap: Adaptive DVFS and Thread Packing Under Power Caps [C]//Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture. New York, USA: ACM Press, 2011:175-185.
- [16] Teodorescu R, Torrellas J. Variation-aware Application Scheduling and Power Management for Chip Multiprocessors [C]//Proceedings of the 35th Annual International Symposium on Computer Architecture. New York, USA: ACM Press, 2008:363-374.
- [17] 王桂彬,杜静,唐滔,等.一种面向异构并行系统的最大功耗管理方法 [J]. 软件学报, 2013, 24(10):2460-2472.
- [18] Ge Rong, Feng Xizhou, Song Shuaiwen, et al. Powerpack: Energy Profiling and Analysis of High-performance Systems and Applications [J]. IEEE Transactions on Parallel and Distributed Systems, 2010, 21(5):658-671.
- [19] Wang Xiaorui, Chen Ming. Cluster-level Feedback Power Control for Performance Optimization [C]//Proceedings of the 14th International Symposium on High Performance Computer Architecture. Washington D. C., USA: IEEE Press, 2008:101-110.
- [20] Spiliopoulos V, Kaxiras S, Keramidas G. Green Governors: A Framework for Continuously Adaptive DVFS [C]//Proceedings of 2011 IEEE International Green Computing Conference and Workshops. Washington D. C., USA: IEEE Press, 2011:1-8.
- [21] Bhattacharjee A, Martonosi M. Thread Criticality Predictors for Dynamic Performance, Power, and Resource Management in Chip Multiprocessors [C]//Proceedings of the 36th Annual International Symposium on Computer Architecture. New York, USA: ACM Press, 2009:290-301.