软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(4):996−1009 [doi: 10.3724/SP.J.1001.2012.04054]

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

多核系统中基于 Global EDF 的在线节能实时调度算法*

张冬松[1+], 吴 彤[2], 陈芳园[1], 金士尧[1]

[1](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)

[2](国防科学技术大学 国家安全与军事战略研究中心,湖南 长沙 410073)

Global EDF-Based On-Line Energy-Aware Real-Time Scheduling Algorithm in Multi-Core Systems

ZHANG Dong-Song[1+], WU Tong[2], CHEN Fang-Yuan[1], JIN Shi-Yao[1]

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

[2](Center for National Security and Strategic Studies, National University of Defense Technology, Changsha 410073, China)

+ Corresponding author: E-mail: dszhang@nudt.edu.cn

Zhang DS, Wu T, Chen FY, Jin SY. Global EDF-based on-line energy-aware real-time scheduling algorithm in multi-core systems. *Journal of Software*, 2012,23(4):996−1009. http://www.jos.org.cn/1000- 9825/ 4054.htm

Abstract: As the energy consumption of multi-core systems becomes increasingly prominent, meeting the time constraints while reducing as much as possible system energy consumption is still an urgent problem in real-time energy-aware scheduling in multi-core system. Most existing works have assumptions on priori information of real-time tasks, but in real applications the tasks' property be received only when the tasks arrive. Therefore, based on the general task model with no priori to tasks' properties, this paper proposes a global EDF-based on-line energy-aware scheduling algorithm for hard real-time tasks in multi-core system. The proposed algorithm can reduce the execution speed of task in multi-core system and reach a reasonable compromise between real-time constraints and energy savings, as it introduces a speed scale factor for utilizing the slack time and combines dynamic power management with dynamic voltage/frequency scaling techniques. The algorithm implements dynamic voltage/frequency scaling only in each context switch time and task completion time, with the smaller computational complexity, and easier to be included in real-time operating system. Experimental results show that the algorithm can be well applied to different kinds of dynamic voltage/frequency scaling on chip, and compared with Global EDF algorithm, it gain more energy savings in all cases, which can improve energy savings about 15% to 20% at most and about 5% to 10% at least.

Key words: real-time system; multi-core system; energy-aware scheduling; dynamic voltage/frequency scaling

摘 要: 随着多核系统能耗问题日益突出,在满足时间约束条件下降低系统能耗成为多核实时节能调度研究中亟待解决的问题之一.现有研究成果基于事先已知实时任务属性的假设,而实际应用中,只有当任务到达之后才能够获

* 基金项目: 国家教育部博士点基金(20104307110005); 湖南省优秀研究生创新资助(CX2010B026); 国防科学技术大学优秀研究生创新资助(B100601)

收稿时间: 2010-07-23; 修改时间: 2011-03-18; 定稿时间: 2011-05-18

CNKI 网络优先出版: 2011-07-04 15:36, http://www.cnki.net/kcms/detail/11.2560.TP.20110704.1536.001.html

得其属性.为此,针对一般任务模型,不基于任何先验知识,提出一种多核系统中基于 Global EDF 在线节能硬实时任务调度算法,通过引入速度调节因子,利用松弛时间,结合动态功耗管理和动态电压/频率调节技术,降低多核系统中任务的执行速度,达到实时约束与能耗节余之间的合理折衷.所提出的算法仅在上下文切换和任务完成时进行动态电压/频率调节,计算复杂度小,易于在实时操作系统中实现.实验结果表明,该算法适用于不同类型的片上动态电压/频率调节技术,节能效果始终优于 Global EDF 算法,最多可节能 15%~20%,最少可节能 5%~10%.

关键词: 实时系统;多核系统;节能调度;动态电压/频率调节

中图法分类号: TP316 **文献标识码:** A

在很多嵌入式实时系统中,特别是无线移动和便携式计算设备,能耗是首要考虑的因素.降低系统运行时的能耗不仅能够延长电池驱动的嵌入式设备的使用,减少体积和重量,而且会因为散热的减少而降低相应的冷却成本,改善性能,延长设备寿命,提高系统可靠性.为了满足嵌入式实时应用的进一步发展,节能调度技术研究已经成为近十几年来学术界和工业界所共同关注的热点之一,并且随着"绿色计算"需求的提出,变得越来越重要.动态功耗管理(dynamic power management,简称 DPM)[1]和动态电压频率调节(dynamic voltage frequency scaling,简称 DVFS)[2]等硬件节能技术在现代多核处理器中的广泛应用,尤其是片上 DVFS 效能的显著提高[3],使得在处理器核内进行更细粒度的能量管理已成为现实,从而为软件层面上提出多核系统中实时节能调度算法带来了可能.

DPM 和 DVFS 通过动态调节系统的运行模式以及降低供应电压和操作频率,实现能量节余.但是这两种技术并没有考虑实时性,很可能会因为执行时间的增加而违背任务的时限约束[4].因此,在多核处理器已成为主流的今天,如何在满足时间约束的前提下尽可能地降低系统能耗,仍是多核实时节能调度研究中一个亟待解决的问题.

许多学者都在研究多处理器或多核处理器系统的实时节能调度问题,但很多研究成果都是在划分法调度的基础上,利用每个处理器上的 DVFS 技术,针对周期或基于帧的任务模型而提出多处理器系统实时节能调度算法[5−7].近年来,针对多处理器系统,一些研究成果开始考虑基于全局调度法的 DVFS 节能调度技术.文献[8]解决了周期性实时任务在同构多处理器上使用 Global EDF 调度的节能调度问题.文献[9]针对周期任务集,提出了基于最优全局实时调度算法 LLREF[10]的多处理器最优离线节能调度算法.文献[11]针对偶发任务集,利用一个全局辅助队列,提出了一种基于 Global EDF 的多处理器在线节能调度算法.

此外,还有不少研究者开始关注多核处理器系统的实时节能调度.针对具有片上统一 DVFS 的同构多核处理器系统,Yang 等人[12]首次提出一种基于划分法的实时节能调度方法,证明多核系统中周期任务的节能调度是一个 NP-hard 问题.Bautista 等人[13]提出一种软实时功耗敏感调度算法,利用片上统一 DVFS 技术,实现软实时任务的节能调度.为了满足多核系统中硬实时约束,Huang 等人[14]提出了一种基于全局固定任务优先级的硬实时节能调度算法.Efraim 等人[15]通过研究多核处理器系统中各种硬件节能技术特点,指出不存在一种算法能够满足所有的情况和约束,各种节能调度算法都有其不同的适用条件.目前,针对异构多核系统的研究主要关注基于启发式的实时节能调度方法.Pepijn 等人[16]针对泄露功耗占主导的异构多核系统,利用 DVFS 和 DPM 提出一系列启发式软实时节能调度算法,但没有考虑硬实时要求.

虽然针对多处理器或多核系统已经有了一些实时节能调度算法,但是目前提出的绝大多数算法一般都假设周期任务模型或偶发任务模型,事先已知任务的到达时间、周期和时限等属性,然后在已知先验知识条件下考虑实时节能调度问题.而在实际的实时应用中,很多情况下无法事先获得任务属性,使得这些算法或将不再有效.为此,本文针对硬实时任务,提出一种多核系统中基于 Global EDF 在线节能调度算法 GEDF-OLEASA(Global EDF-based on-line energy-aware scheduling algorithm).该算法利用片上 DVFS 和 DPM,降低动态功耗和泄露功耗,达到实时约束与能耗节余之间的合理折衷.GEDF-OLEASA 算法考虑了一般任务模型,每个任务具有任意释放时间和时限,不再假设事先已知实时任务的任何属性,只有当任务到达之后才能够获得其具体属性.同时,在任务分配问题的求解上,本文也不同于现有大多数算法.本文不是基于划分法实现任务调度,而是采用基于

Global EDF 的任务调度方法[17],任务集中每个任务实例的优先级在运行时固定.虽然基于划分法的任务调度易于实现单个核上的节能调度,但是最优的划分方法是典型的 NP 难问题.相比之下,全局法调度更适合动态系统,当任务集发生变化时不需要运行负载均衡或任务分配算法.本文所提出的算法只在任务上下文切换时刻和任务完成时刻实现动态电压/频率调节,计算复杂度小,易于在实时操作系统中实现.

本文第 1 节介绍系统模型,并给出问题的定义.第 2 节详细描述 GEDF-OLEASA 算法.算法的理论分析和证明在第 3 节中给出.第 4 节通过实验分析算法的节能效果,并与 Global EDF 算法进行比较.第 5 节对全文作简要总结.

1 系统模型与问题定义

这一节提出本文所用的处理器模型、任务模型以及能耗模型,并基于模型给出问题定义.

1.1 处理器模型

本文假设片上多核处理器具有 m 个同构处理器核,m 为常数,给定的处理器核命名为 $core_1,core_2,\ldots,core_m$.处理器核 $core_i$ 以速度 S 运行时的功耗 $P_i(S)$为

$$P_i(S)=P_i^{dep}(S)+P_i^{ind}.$$ [17]

$P_i^{dep}(S)$ 和 P_i^{ind} 分别表示与速度相关的功耗和与速度无关的功耗[17]. $P_i^{dep}(S)$ 主要由门电路电容充放电引起的动态功耗(dynamic power)和短路瞬态电流产生的短路功耗(short-circuit power)构成. $P_i^{dep}(S)$ 通常可以作为处理器速度的凸函数.根据文献[12,18–20]可知,与速度相关的功耗 $P_i^{dep}(S)$ 可表示为

$$P_i^{dep}(S)=S^3.$$

与速度无关的功耗 P_i^{ind} 主要来自于漏电流产生的泄露功耗,可设为

$$P_i^{ind}=\beta,$$ [21]

其中,β是一个常量.因此,功耗函数可表示为

$$P_i(S)=S^3+\beta.$$ [18]

这种功耗模型适用于多种 DVFS 处理器,例如 Intel Xscale 处理器的功耗函数可以近似为一个常量加上速度的立方函数[18,22].有很多文献[7,18,19,21]基于上述功耗模型展开研究,受文章篇幅所限,本文不加详述,有关功耗模型的详细说明可参考文献[18].

进一步地,假设同构多核处理器具有 DPM 和片上 DVFS 技术[3].片上 DVFS 分为两类:① 片上统一 DVFS (global DVFS),片上所有核都在相同的时钟域内,它们必须运行在相同的时钟周期和供应电压下,可以设计统一的电压/频率调节策略,例如 ARM 公司的 ARM11 MPCore 处理器[23];② 片上每个核 DVFS(per-core DVFS),片上每个核都在不同的时钟域内,可以分别运行在不同的时钟周期和供应电压下,可以对每个处理器核设计独立的电压/频率调节策略,例如 AMD 公司的 Opteron 多核处理器[24].

1.2 任务模型

本文考虑硬实时系统中一般的实时任务集$\Gamma=\{\tau_1,\tau_2,\ldots,\tau_n\}$,任务 τ_i 用三元组(r_i,D_i,C_i)表示,其中,r_i 是释放时间,D_i 是时限,C_i 是任务 τ_i 在最高频率下的最坏情况执行时间或时钟周期数(worst-case execution time,简称 WCET).如果 τ_i 是周期任务,则定义其周期为 T_i,利用率为 $U_i=C_i/T_i$.如果 τ_i 是非周期任务,则定义其任务密度为 $\lambda_i=C_i/D_i$.假设所有的任务都是独立的,具有任意的释放时间和时限.同时,定义绝对时限 $d_i=r_i+D_i$,s_i 是任务 τ_i 的开始执行时间,c_i 是任务 τ_i 在最坏情况最大完成时间,e_i 是在最高频率下的实际执行时间.

任务调度算法采用最早时限优先(earliest deadline first,简称 EDF)可抢占策略.由于实时任务的实际执行时间 e_i 通常远小于最坏执行时间 C_i,所以可以利用松弛时间动态地降低电压和频率.

1.3 能耗模型

目前,绝大多数商业多核处理器都同时具有片上DVFS技术和DPM功能,因此,本文针对这种典型情况考虑电压/频率级别可以连续调节的理想 DVFS 情况.不妨假设同构多核处理器可以在$[S^{\min},S^{\max}]$速度范围内连续调节,$S^{\min}$和$S^{\max}$分别表示最小和最大处理器速度.令α为速度调节因子,在t时刻处理器的频率降低为$\alpha\cdot S^{\max}$.另外,DPM 启动和关闭处理器核的能耗忽略不计,而时间开销可以考虑加入每个任务的 WCET 中.

根据第 1.1 节功耗函数可知,处理器核 $core_i$ 以速度 S 执行一个时钟周期的能耗为 $P_i(S)/S=S^2+\dfrac{\beta}{S}$, 如图 1 所示.对于具有 DPM 的多核处理器,存在一个关键速度 $S^{critical}$(critical speed),即处理器执行一个时钟周期所用能耗最小的速度[7,18].从图 1 可知,对 $P_i(S)/S$ 求一阶导数 $\mathrm{d}(P_i(S)/S)=2S-\beta\cdot S^{-2}$,使得 $P_i(S)/S$ 为最小值的可用执行速度 S 就是 $S^{critical}=\sqrt[3]{\beta/2}$.显然,$S^{critical}$ 满足如下性质[7,18]:$\forall S, S^{\min}\leqslant S\leqslant S^{\max}, P_i(S^{critical})/S^{critical}\leqslant P_i(S)/S$.

在$(t_1,t_2]$时间间隔内,令处理器核 $core_i$ 的速度调节因子α_i保持不变,$T=t_2-t_1$,则 $core_i$ 的能耗 $E_i(\alpha_i)$为功耗在 T 时间间隔内的积分

$$E_i(\alpha_i)=\int_0^T P_i(S_i)\mathrm{d}t.$$

当$\alpha_i=0$ 时,该处理器核被关闭,能耗显然为 0.当 $0<\alpha_i\leqslant S^{critical}/S^{\max}$ 时,$S_i=\alpha_i\cdot S^{\max}\leqslant S^{critical}$.如果按照速度 S_i 执行 T 个时间单位,根据上述分析可知,因为泄露功耗的影响,该处理器核的能耗不降反升.因此,该处理器核的速度可以设为关键速度 $S^{critical}$,以获得更低的能耗值.同时,由于在 T 时间间隔内执行的 CPU 时钟周期数与处理器速度近似成正比[4,7,18],处理器核的执行时间会缩短为 $T\cdot(\alpha_i S^{\max}/S^{critical})$.当 $S^{critical}/S^{\max}<\alpha_i\leqslant 1$ 时,$S_i=\alpha_i\cdot S^{\max}>S^{critical}$,该处理器核按照速度 S_i 执行,且执行时间刚好为 T.为保证任务的时限要求,处理器核的速度无法降低为最低能耗时的 $S^{critical}$.形式化描述如下:

$$S_i=\begin{cases}0, & \text{if } \alpha_i=0\\ S^{critical}, & \text{if } 0<\alpha_i\leqslant S^{critical}/S^{\max}\\ \alpha_i\cdot S^{\max}, & \text{if } S^{critical}/S^{\max}<\alpha_i\leqslant 1\end{cases},$$

$$T=\begin{cases}0, & \text{if } \alpha_i=0\\ T\cdot(\alpha_i S^{\max}/S^{critical}), & \text{if } 0<\alpha_i\leqslant S^{critical}/S^{\max}\\ T, & \text{if } S^{critical}/S^{\max}<\alpha_i\leqslant 1\end{cases}.$$

最后,根据第 1.1 节中功耗函数可得:

$$E_i(\alpha_i)=\int_0^T P_i(S_i)\mathrm{d}t=\begin{cases}0, & \text{if } \alpha_i=0\\ T\cdot(P_i^{dep}(S^{critical})+P^{ind})\cdot\alpha_i S^{\max}/S^{critical}, & \text{if } 0<\alpha_i\leqslant S^{critical}/S^{\max}\\ T\cdot(P_i^{dep}(\alpha_i\cdot S^{\max})+P_i^{ind}), & \text{if } S^{critical}/S^{\max}<\alpha_i\leqslant 1\end{cases}\quad(1)$$

图 2 给出了具有 DPM 的理想 DVFS 情况下的能耗 $E_i(\alpha_i)$,其中,$P^{dep}(S_i)=(S_i)^3$,$P^{ind}=\beta$, $S^{critical}=\sqrt[3]{\beta/2}$,$\beta$是常量.

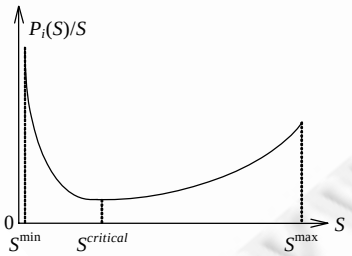


Fig.1 An example for function $P_i(S)/S$

图 1 $P_i(S)/S$ 函数示例

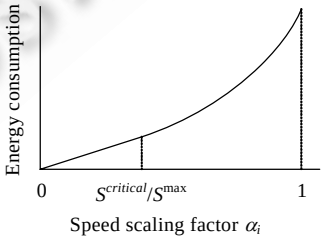


Fig.2 Ideal DVFS with DPM

图 2 具有 DPM 的理想 DVFS

1.4 问题定义

给定一般的实时任务集Γ,可以在由 m 个处理器核组成的同构多核处理器上运行.假设事先不知道实时任务的任何属性,具体属性只有在任务到达后才可获得.假设实时任务集是可调度的,每个任务的最坏情况完成时间都不超过其时限.同时,假设同构多核处理器均具有DVFS和DPM功能,不考虑启动和关闭处理器核的能耗和时间开销,且速度切换开销忽略不计.

本文研究的目标是,在保证实时任务集满足最坏情况可调度性的前提下,尽可能地降低系统能耗.因此根据公式(1),问题形式化定义如下:

- Minimize $\sum_{i=1}^{m} E_i(\alpha_i)$
- Subject to $E_i(\alpha_i)=E_i(P_i(S_i))=\int_0^T P_i(S_i)\mathrm{d}t,\ T>0, i\in\{1,2,\ldots,m\}$

  $P_i(S_i)=(S_i)^3+\beta, i\in\{1,2,\ldots,m\}, \beta$是常数

  $\forall i\in\{1,2,\ldots,n\}, c_i\leqslant d_i$

2 GEDF-OLEASA 算法

本节首先介绍 GEDF-OLEASA 算法中涉及到的基本概念,然后提出基本思想,最后给出具体描述以及复杂度分析.

2.1 基本概念

偶发任务是非周期任务的特例[25],本文将偶发任务的每个任务实例看作单独的非周期任务.我们将文献[26]提出的基于 Global EDF 的偶发任务集可调度性判断策略扩展到一般实时任务集,给出判断任务集可调度性的充分条件,具体如下:

① 问题窗口是指一个出现时限丢失的时间间隔,以时限丢失时刻为结束点[26].例如,任务τ_i从释放时刻 r_i 到丢失时限时刻 d_i 之间的时间间隔如图 3 所示;

② 时限丢失的必要条件:对于任务τ_i,所有 m 个处理器在整个问题窗口中执行其他任务的时间大于 D_i-C_i;

③ 根据步骤①和步骤②,得到任务τ_i 在这个时间间隔内由于其他任务执行而产生的最大干扰时间上界 I^{UB}.其他任务包括在该间隔内释放的任务以及在间隔开始之前已经释放但未完成的任务;而任务τ_i 的干扰时间是指 m 个处理器被优先级高于任务τ_i的任务或任务实例占用的时间;

④ 根据步骤②和步骤③,形成一个必要的不可调度性测试,如 $I^{UB}>D_i-C_i$;然后对不等式取反,形成一个充分的可调度性测试,即 $I^{UB}\leqslant D_i-C_i$.

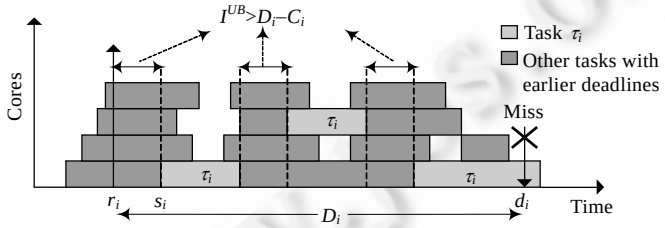


Fig.3 Problem window

图 3 问题窗口

借鉴上述问题窗口的思想,本文提出了任务τ_i的最坏情况最大完成时间 c_i 这一概念,即

$$c_i=s_i+C_i+I_i(d_i-s_i), r_i\leqslant s_i<d_i \tag{2}$$

其中,$I_i(d_i-s_i)$是任务τ_i在$[s_i,d_i]$内由于其他高优先级任务执行而产生的干扰时间.由图 3 可知,公式(2)中干扰任务

τ_i 的其他高优先级任务主要有两类:一是在$[s_i,d_i]$内释放且优先级高于 τ_i 的任务,形式化描述为$\forall j,s_i\leqslant r_j,d_j\leqslant d_i$;二是输入(carry-in)任务,它是在 s_i 之前已释放但在 s_i 时刻还未完成的任务,形式化描述为$\forall j,r_j<s_i,s_i<d_j\leqslant d_i$.

进一步地,本文引入任务集的可调度性定义.

定义 1. 在给定的调度算法中,一个任务称为可调度的,是指它在该调度算法下的最坏情况完成时间小于或等于它的时限.类似地,在给定的调度算法中,一个任务集称为可调度的,是指它的所有任务在该算法下都是可调度的[17].

2.2 基本思想

针对由周期任务和非周期任务构成的一般实时任务集,由于事先不知道任务的任何属性,本文在任务调度时必须假设任务执行在最坏情况,即每个任务的实际执行时间等于 WCET.此时,任务集在实时调度中没有动态松弛时间.为满足硬实时要求,不能降低多核处理器的执行速度.但是,任务的实际执行时间通常远小于 WCET,如果仍然以最高速度运行该任务,那么任务的实际最大完成时间 K_i 必然小于最坏情况最大完成时间 c_i.这样,由于前面任务较早完成而产生的动态松弛时间就为降低后续任务的执行速度提供了可能性.

EDF 算法是单处理器系统中经典的最优动态优先级调度算法,但在多处理器或多核系统中并不是最优的.而在多处理器或多核系统中,虽然对于周期任务集合存在最优的全局调度法,但是对于非周期任务集,目前还没有发现一种最优全局调度法.而且,如果不知道非周期任务的到达时间,最优的全局调度法也不可能获得[17].因此,本文需要假设一般任务集在可抢占的Global EDF 调度算法中是可调度的,即由定义 1 可知,在该调度算法中,每个任务 τ_i 都必须在 c_i 时刻或之前完成,即 $K_i\leqslant c_i\leqslant d_i$.显然,如果能够设计一种在线节能调度算法,当每个任务的实际执行时间在不超过 WCET 时,保证任务总能在 c_i 之前完成,那么该算法就能满足可调度性.根据公式(2)中关于 c_i 的计算以及任务干扰时间的定义,本文考虑在任务上下文切换和任务完成时,引入任务的执行速度调节因子 α_i 的计算,进而实现动态电压/频率调节.具体过程如下:

① 如果任务 τ_i 在 t 时刻抢占第 k 个处理器核上的任务 τ_j,则 $K_i=t+C_i$;

② 如果任务 τ_i 在第 k 个处理器核上的任务 τ_j 完成之后恢复执行,那么 $K_i=K_i+K_{\min}-t_{ip}$.这里,$K_{\min}$ 是所有当前正在运行任务的最大完成时间 K_i 的最小值,t_{ip} 是上一次抢占时刻;

③ 如果任务 τ_i 在第 k 个核上的任务 τ_j 完成后首次执行,当任务 τ_i 的时限 d_i 不小于当前在 m 个处理器核上运行任务的最大时限 $d_{\max}$,且存在来自其他高优先级任务的干扰,即 $K_{\min}\geqslant t$ 时,则 $K_i=K_{\min}+C_i$;否则,$K_i=t+C_i$;

④ 当得到每个任务的当前最大完成时间后,还需要更新 $d_{\max}$ 和 $K_{\min}$.此时不仅需要更新当前在第 k 个处理器核上运行任务的时限 d_k 以及 $d_{\max}=\max\{d_k\},k\in[1,m]$,还需要更新当前在第 k 个处理器核上运行任务的最大完成时间 K_k 以及 $K_{\min}=\min\{K_i\},k\in[1,m]$;

⑤ 当任务 τ_i 首次执行时,由于事先无法获知其实际执行时间,必须假设任务 τ_i 在最坏情况执行,即任务 τ_i 的剩余最大执行时钟周期数 R_i 等于 C_i.在任务执行期间 R_i 不断减少,任务 τ_i 完成时 R_i 为 0.如果任务 τ_j 在 t 时刻被其他高优先级任务抢占,那么需要更改任务 τ_i 的剩余最大执行时间,即 $R_j=R_j-\alpha_j(t-l_k)$,其中,l_k 是上一次第 k 个核上的上下文切换时刻.因此,当任务 τ_i 在 t 时刻被上下文切换到第 k 个处理器核时,速度调节因子可以设为 $\alpha_i=R_i/(K_i-t)$;

⑥ 除了在上下文切换时刻需要计算速度调节因子以外,当任务 τ_i 在第 k 个处理器核完成时,如果此时就绪队列为空,没有任务等待执行,则 $\alpha_i=0$,重新进行动态电压/频率调节.

2.3 算法描述

虽然实时任务均采用最坏执行时间 C_i 来描述,但实际上很少出现最坏情况的数值,任务在执行时会产生松弛时间.因此,当任务产生松弛时间后,多核处理器中 DVFS 技术会根据松弛时间来动态调节处理器的电压和频率,在满足硬实时约束条件下降低系统能耗.图 4 给出了 GEDF-OLEASA 算法描述.*Calculate_*$\alpha(\tau_i,t,k)$函数分析了当前任务的不同调度类型,根据抢占、恢复执行和非抢占等不同情况来计算任务的当前最大完成时间 K_i,并保存最小值 $K_{\min}$ 以获取当前最大干扰的上界,进而可以得到任务调度执行时的速度调节因子,实现了

GEDF-OLEASA 算法中速度调节因子的计算过程.

Upon Context Switch to Task τ_i at Time t in the kth Core:
1: α_i=*Calculate_α*(τ_i,t,k);
2: *Scale_Voltage_and_Frequency_All_Core*(α_i,k);
or *Scale_Voltage_and_Frequency_Each_Core*(α_i,k);

***Calculate_α*(τ_i,t,k)**:
1: if τ_i preempted τ_j in the kth core then /*任务τ_i释放后在第 k 个核上抢占τ_j执行*/
2: {$K_i=t+C_i$;
3: $R_i=C_i$;
4: $R_j=R_j-\alpha_k(t-l_k)$;
5: }
6: else if τ_i resumes after some task τ_j in the kth core then
7: $K_i=K_i+K_{\min}-t_{ip}$; /*任务τ_i在t_{ip}时刻被抢占过*/
8: else /*任务τ_i释放后在第 k 个核上首次非抢占执行*/
9: {if ($d_{\max}\leqslant d_i$ and $K_{\min}\geqslant t$) then
10: $K_i=K_{\min}+C_i$;
11: else $K_i=t+C_i$;
12: $R_i=C_i$;
13: }
14: update (d_k,$d_{\max}$); /*初始化 d_k=0,$d_{\max}$=0;更新操作:$d_k=d_i$, if ($d_k>d_{\max}$) then $d_{\max}=d_k$;*/
15: update (K_k,$K_{\min}$); /*初始化 K_k=0,$K_{\min}=K_k$;更新操作:$K_k=K_i$, if ($K_k<K_{\min}$) then $K_{\min}=K_k$;*/
16: return ($\alpha_i=R_i/(K_i-t)$); /*返回速度调节因子α_i*/

***Upon_Task_Completion*(τ_i) in the kth Core:**
1: R_i=0;
2: if 就绪队列为空 then
3: {α_i=0;
4: *Scale_Voltage_and_Frequency_All_Core*(α_i,k);
or *Scale_Voltage_and_Frequency_Each_Core*(α_i,k);
5: }

***Scale_Voltage_and_Frequency_All_Core*(α_i,k)**:
1: $\alpha_k=\alpha_i$;
2: if α_k=0 then 关闭第 k 个处理器核;
3: 从 m 个处理器核的速度调节因子中选取最大值 $\alpha_{\max}=\max_{\forall k\in\{1,2,\ldots,m\}}\{\alpha_k\}$;
4: if ($\alpha_{\max}\neq 0$ and $\alpha_{\max}<S^{critical}/S^{\max}$) then $\alpha_{\max}=S^{critical}/S^{\max}$;
5: 对所有$\alpha_k\neq 0$ 的处理器核,$S=\alpha_{\max}\cdot S^{\max}$;

***Scale_Voltage_and_Frequency_Each_Core*(α_i,k)**:
1: $\alpha_k=\alpha_i$;
2: if α_k=0 then 关闭第 k 个处理器核;
3: if ($\alpha_k\neq 0$ and $\alpha_k<S^{critical}/S^{\max}$) then $\alpha_k=S^{critical}/S^{\max}$;
4: 对所有$\alpha_k\neq 0$ 的处理器核,$S=\alpha_k\cdot S^{\max}$;

Fig.4 Pesudo code of GEDF-OLEASA algorithm

图 4 GEDF-OLEASA 算法伪代码描述

当速度调节因子确定以后,该算法根据多核处理器本身所具备的不同 DVFS 和 DPM 技术,对处理器核实现动态电压/频率调节,在保证实时任务集可调度性的条件下,合理地降低系统能耗.在任务的上下文切换时刻实现动态调节处理器核的电压/频率,保证任务的实际完成时间不超过任务在最坏情况下的最大完成时间,既满足了 GEDF-OLEASA 算法的可调度性,又实现了节能设计的要求.同时,在任务完成时刻,当就绪队列为空时,也需要改变速度调节因子,这样可以在满足当前实时任务可调度性的条件下进一步实现节能.

Scale_Voltage_and_Frequency_All_Core(α_i,k)函数针对具有片上统一 DVFS 和 DPM 的多核处理器,通过选取当前所有处理器核上最高速度调节因子来实现节能;而 *Scale_Voltage_and_Frequency_Each_Core*(α_i,k)函数则针对具有片上每个核 DVFS 和 DPM 的多核处理器,在每个处理器核上实现独立的电压和频率动态调节功能.

2.4 复杂度分析

设全局就绪队列长度为 n,处理器核数为 m,每个任务状态数量为 w.

时间复杂度:在计算速度调节因子的函数中,计算 K_i 和 R_i 的时间复杂度为常数,记作 $O(1)$;如果采用存储在

顺序表上的顺序查找法,则更新 d_k 和 $d_{\max}$ 的时间复杂度为 $O(1)+O(m)$;但是,如果采用存储在完全二叉树上的锦标赛排序法,则更新 d_k 和 $d_{\max}$ 的时间复杂度可以降低为 $O(1)+O(\log_2 m)$;同理,更新 K_k 和 $K_{\min}$ 的时间复杂度也可降低为 $O(1)+O(\log_2 m)$.于是,计算速度调节因子的时间复杂度为 $O(\log_2 m)$.另外,由于采用锦标赛排序法的选择最大速度调节因子的时间复杂度也可以降低为 $O(\log_2 m)$,因此利用片上统一 DVFS 和 DPM 动态调节电压和频率的时间复杂度为 $O(1)+O(\log_2 m)$;而利用片上每个核 DVFS 和 DPM 动态调节电压和频率的时间复杂度则为 $O(1)$.此外,当任务完成时,动态调节电压和频率在最坏情况下的时间复杂度为 $O(\log_2 m)$.因此,在任务 τ_i 的上下文切换时刻和完成时刻,进行动态电压和频率调节的时间复杂度为 $O(\log_2 m)+(O(1)+O(\log_2 m))+O(\log_2 m)=O(\log_2 m)$,或者 $O(\log_2 m)+O(1)+O(\log_2 m)=O(\log_2 m)$.

空间复杂度:任务集 Γ 所占用的空间为 $O(n\cdot w)$.处理器状态包括 d_k,K_k 以及速度调节因子 α_k 等.如果采用顺序表存储,每个处理器状态所占用的空间为 $O(m)$;如果采用完全二叉树存储,每个处理器状态所占用的空间为

$$O(2\cdot 2^{\lceil \log_2 m \rceil}-1).$$

3 算法分析

本节首先证明如下 3 个引理,在此基础上,证明 GEDF-OLEASA 算法的可调度性.

引理 1. 如果已知实时任务集在 Global EDF 调度算法下可调度,那么任务的最坏情况完成时间就是 Global EDF 最坏情况调度序列中对应的任务完成时间.

证明:由定义 1 可知,已知实时任务集在 Global EDF 调度算法下是可调度的,则说明在该调度算法下每个任务的完成时间都不会超过其时限.显然,在最坏情况时,即每个任务的实际执行时间都等于 WCET,该任务集中所有任务在 Global EDF 调度算法下的完成时间也不会超过任务的时限.此外,由于本文假设事先不知道实时任务的任何属性,只有当任务释放时才能得到任务属性,所以,每当任务被调度执行时,必须假设任务在最坏情况执行.因此,每个任务的最坏情况完成时间就等于在 Global EDF 最坏情况调度序列中对应的完成时间. □

引理 2. 如果每个实时任务都在最坏情况执行,那么 GEDF-OLEASA 算法中所计算的最大完成时间 K_i 就等于 Global EDF 调度算法中最坏情况完成时间.

证明:如果每个任务都在最坏情况执行,即每个任务的实际执行时间都等于其 WCET.由于 GEDF-OLEASA 算法中速度调节因子的改变只在任务的上下文切换时刻和完成时刻发生,所以,首先根据不同调度类型,分 4 种情况来说明上下文切换时速度调节因子的计算,然后分析任务完成时刻速度调节因子的变化.

第 1 种情况:当出现任务 τ_i 抢占第 k 个处理器核上的任务 τ_j 时,任务 τ_i 的剩余最大时钟周期数 R_i 就等于 C_i,且所计算的最大完成时间 K_i 等于 $t+C_i$.显然,此时得到的速度调节因子 $\alpha_i=1$;

第 2 种情况:若任务 τ_i 释放时有空闲处理器核,则 τ_i 在空闲核上首次开始执行.此时,虽然不能确定任务 τ_i 的优先级一定低于当前正在运行的任务的优先级,但是所有正在运行任务的 K_i 都会超过当前时刻,而且已完成任务的实际完成时间都等于其最大完成时间,不存在 K_i 的最小值小于当前时刻的情况.因此,τ_i 的当前最大完成时间 $K_i=t+C_i$,$R_i=C_i$,从而有速度调节因子 $\alpha_i=1$;

第 3 种情况:任务 τ_i 释放时没有执行,此时首次开始执行.这说明任务 τ_i 释放时优先级低于 m 个处理器核上当时正在运行任务的优先级.由于已知每个任务的实际执行时间都等于 WCET,所以该任务必定等到某个高优先级任务执行完成后之才可以开始执行;并且,此时前一个已完成任务 τ_j 的最大完成时间就等于当前时间.而 τ_i 的当前最大完成时间 $K_i=t+C_i$,$R_i=C_i$,从而有速度调节因子 $\alpha_i=1$;

第 4 种情况:任务 τ_i 之前被抢占过,现在是在高优先级任务 τ_j 完成之后恢复执行.由于已知每个任务的执行时间都等于 WCET,并且抢占执行和非抢占执行的任务速度调节因子都等于 1,没有速度调节,所以,任务 τ_j 的完成时间就等于最大完成时间;再根据 GEDF-OLEASA 算法计算得到任务 τ_i 的速度调节因子 $\alpha_i=1$.

当任务完成时,GEDF-OLEASA 算法从就绪队列中选择优先级最高的任务进行调度.如果此时就绪队列中没有等待任务,这时,当前 m 个处理器核中必然会出现空闲核.为了进一步节能,将关闭空闲核,并重新设置其他非空闲核的速度调节因子.对于具有片上统一 DVFS 的多核系统,由于之前每个核上正在运行任务的速度调节

因子均为 1,所以重新计算的最大速度调节因子仍是 1,保持不变.对于具有片上每个核 DVFS 的多核系统,不需要计算最大速度调节因子,每个核的速度调节因子仍然保持不变.

综上所述,GEDF-OLEASA 算法所计算的速度调节因子总为 1.显然,在该算法中没有实现速度调节.这样,GEDF-OLEASA 算法的调度序列等同于实时调度算法的调度序列.此时,由引理 1 可知,GEDF-OLEASA 算法中所计算的最大完成时间 $K_i^{\alpha_i=1}$ 就等于 Global EDF 实时调度算法中最坏情况完成时间. □

引理 3. 如果不是所有任务都在最坏情况下执行,那么 GEDF-OLEASA 算法中所计算的最大完成时间 K_i 不会超过 Global EDF 实时调度算法中最坏情况完成时间,而且每个任务的实际完成时间均不超过计算的最大完成时间 K_i.

证明:如果不是所有任务都在最坏情况下执行,则存在某些任务的实际执行时间小于 WCET.GEDF-OLEASA 算法中速度调节因子的改变只能在任务的上下文切换时刻和完成时刻发生.

对于上下文切换时刻,本文根据不同的任务调度类型,分情况来证明 GEDF-OLEASA 算法中所计算的每个任务的最大完成时间 K_i 不会超过最坏情况该算法所计算的最大完成时间 $K_i^{\alpha_i=1}$, 说明每个任务的实际完成时间不会超过 K_i:

第 1 种情况:当任务 τ_i 抢占第 k 个处理器核上的任务 τ_j 时,根据 GEDF-OLEASA 算法,显然可以得到速度调节因子 α_i=1,所以 τ_i 的最大完成时间 K_i 就等于最坏情况下的最大完成时间 $K_i^{\alpha_i=1}$;

第 2 种情况:任务 τ_i 在第 k 个核上首次非抢占执行.因为每个任务的实际执行时间可以小于 WCET,所以又可以细分为 3 种情况:

① 如果任务 τ_i 的时限小于当前在 m 个处理器核上正在运行任务的最大时限 $d_{\max}$,即 $d_i<d_{\max}$,那么说明任务 τ_i 在 Global EDF 最坏情况的调度序列中会抢占某个任务,因此,速度调节因子 α_i=1.显然, $K_i = K_i^{\alpha_i=1}$;

② 如果任务 τ_i 的时限大于或等于当前在 m 个处理器核上运行任务的最大时限 $d_{\max}$,即 $d_{\max}\leqslant d_i$,同时,当前所保存的正在运行任务最大完成时间的最小值 $K_{\min}$ 不小于当前时间 t,则说明多核处理器上已完成的任务未按照最坏情况运行,任务 τ_i 提前执行,而 $K_{\min}-t$ 就是最坏情况时任务 τ_i 所受到的来自其他高优先级任务的干扰时间上界,它可以被作为松弛时间用于计算速度调节因子,降低处理器速度.因为任务 τ_i 的当前最大完成时间 K_i 是按照当前最小值 $K_{\min}$ 计算得到的,而不是按照最坏情况调度队列中的前一个已完成任务的最大完成时间 $K_j(K_j\geqslant K_{\min})$计算得到,所以,GEDF-OLEASA 算法所计算的每个任务的当前最大完成时间 K_i 不会大于最坏情况下的最大完成时间 $K_i^{\alpha_i=1}$;

③ 如果 GEDF-OLEASA 算法调度任务 τ_i 时,当前正在执行任务的最大完成时间的最小值 $K_{\min}$ 小于当前时间 t,这说明,在 t 之前一直有处理器核处于空闲状态,任务 τ_i 在 Global EDF 最坏情况下的调度序列中也没有抢占执行,此时,速度调节因子 α_i=1.显然, $K_i = K_i^{\alpha_i=1}$.

第 3 种情况:当任务 τ_i 之前被抢占过时,由于高优先级任务 τ_j 的完成而恢复执行.此时,GEDF-OLEASA 算法是按照当前最小值 $K_{\min}$ 减去前一个抢占时刻来计算任务 τ_i 的最大完成时间 K_i.同理,因为不是按照最坏情况下调度队列中的前一个已完成任务的最大完成时间 $K_j(K_j\geqslant K_{\min})$计算得到的,因此 $K_i\leqslant K_i^{\alpha_i=1}$.

当任务的实际执行时间小于 WCET 时,通过 K_i 计算得到的每个任务的速度调节因子有可能小于 1;但根据计算公式 $\alpha_i=R_i/(K_i-t)$可知,该速度调节因子能够保证任务的实际完成时间不超过 K_i.那么,在进行动态电压/频率调节时,对于具有片上统一 DVFS 的多核系统,因为需要选择最大速度调节因子来统一调节所有核的电压/频率级别,所以每个核上正在运行任务的实际执行速度只会大于计算的速度值,任务的实际执行时间也不会超过 K_i.而对于具有片上每个核 DVFS 的多核系统,按照每个核的当前速度调节因子分别设置各自的电压和频率,所以每个核上正在运行任务的实际执行速度就等于计算的速度值,显然,任务的实际执行时间不超过 K_i.

对于任务完成时刻,GEDF-OLEASA 算法会从就绪队列中选择优先级最高的任务进行调度.如果就绪队列中存在任务,则进入任务调度的上下文切换时刻.但如果此时就绪队列中没有任务,当前 m 个处理器核中必然会出现空闲核,为了进一步节能将关闭空闲核,并重新设置其他非空闲核的速度调节因子.此时,对于具有片上每个核 DVFS 的多核系统,不需要计算最大速度调节因子,每个核的速度调节因子仍然保持不变,每个核上正在运

行任务的 K_i 和实际完成时间均保持不变.对于具有片上统一 DVFS 的多核系统,由于之前每个非空闲核上正在运行任务的速度调节因子没有改变,所以从非空闲核中重新计算的最大速度调节因子不会超过原来的值,每个任务的实际执行时间也不会超过 K_i.

综上所述,GEDF-OLEASA 算法所计算的每个任务的最大完成时间 K_i 不会超过最坏情况下最大完成时间 $K_i^{\alpha_i=1}$, 而且每个任务的实际完成时间均不超过最大完成时间 K_i.又由引理 2 可知 K_i 不会超过 Global EDF 实时调度算法中最坏情况完成时间. □

定理 1. 每个在 Global EDF 调度算法下可调度的实时任务集在 GEDF-OLEASA 算法下也是可调度的.

证明:由引理 2 和引理 3 可知,GEDF-OLEASA 算法中每个任务的最大完成时间 K_i 不会超过 Global EDF 实时调度算法中最坏情况完成时间,而且每个任务的实际完成时间不超过 K_i.同时,又已知实时任务集在 Global EDF 中是可调度的,所以任务在 GEDF-OLEASA 下的实际完成时间不会超过其时限.命题得证. □

由于任务的实际执行时间通常都低于其 WCET,因此 GEDF-OLEASA 算法可以利用任务提前完成产生的动态松弛时间,延长后续任务的执行时间,进而达到节能的目标,定理 2 和定理 3 分别给出最坏情况和一般情况下的节能特性.

定理 2. 最坏情况下,GEDF-OLEASA 不会比 Global EDF 更耗能.

证明:最坏情况就是所有任务的实际执行时间都等于其 WCET.由引理 2 可知,在 GEDF-OLEASA 算法中,任务 τ_i 在第 k 个处理器核上以 $S^{\max}$ 执行 C_i 时间,在 Global EDF 算法中同样在第 k 个核上以 $S^{\max}$ 执行 C_i 时间.根据公式(1),显然,在两种算法下,每个任务的能耗均相等,且每个核的能耗也相等. □

定理 3. 一般情况下,GEDF-OLEASA 始终优于 Global EDF.

证明:一般情况是指不是所有任务都在最坏情况下执行,即存在某个任务的实际执行时间小于其 WCET.由引理 3 可知,在 GEDF-OLEASA 算法中,任务 τ_i 在第 k_1 个处理器核上以 S_1 执行 t_1 时间,而在 Global EDF 算法中,任务 τ_i 在第 k_2 个处理器核上以 $S^{\max}$ 执行 t_2 时间.

这里,k_1 和 k_2 可相同也可不同,$t_1 \leqslant t_2$,$S^{critical} \leqslant S_1 \leqslant S^{\max}$,$S_1 \cdot t_1 = S^{\max} \cdot t_2$.

根据公式(1),在 GEDF-OLEASA 算法中,任务 τ_i 耗能 $E_1=((S_1)^3+\beta)\cdot t_1$;

在 Global EDF 算法中,任务 τ_i 耗能 $E_2=((S^{\max})^3+\beta)\cdot t_2$,两式相除,并将 $S_1 \cdot t_1 = S^{\max} \cdot t_2$ 代入得到:

$$\frac{E_1}{E_2}=\frac{((S_1)^3+\beta)\cdot t_1}{((S^{\max})^3+\beta)\cdot t_2}=\frac{((S_1)^3+\beta)\cdot S^{\max}}{((S^{\max})^3+\beta)\cdot S_1}=\frac{(S_1)^2+\beta/S_1}{(S^{\max})^2+\beta/S^{\max}}.$$

因为 $S^{critical} \leqslant S_1 \leqslant S^{\max}$,根据公式(1)中能耗函数递减特性可知,$P_i(S_1)/S_1 \leqslant P_i(S^{\max})/S^{\max}$,即

$$(S_1)^2+\beta/S_1 \leqslant (S^{\max})^2+\beta/S^{\max}.$$

证得 $\frac{E_1}{E_2} \leqslant 1$, 即 GEDF-OLEASA 节能效果始终优于 Global EDF. □

4 实 验

本节通过实验比较 Global EDF 和 GEDF-OLEASA 算法的性能.其中,根据多核处理器所采用片上 DVFS 技术的不同,GEDF-OLEASA 算法可以分为 GEDF-OLEASA(ALL)和 GEDF-OLEASA(EACH)两种情况,前者是针对具有片上统一 DVFS 和 DPM 的多核处理器,后者是针对片上每个核 DVFS 和 DPM 的多核处理器.

4.1 参数产生

实验选取由 n 个硬实时任务($n>>m$)组成的任务集 Γ,假设任务之间相互独立.对于多核处理器,考虑允许电压/频率连续调节的理想情况.假设 $\beta=0.1\cdot(S^{\max})^3$, $S^{critical}=\sqrt[3]{\beta/2}=0.37\cdot S^{\max}$ [18].根据公式(1)计算能耗,忽略片上 DVFS 和 DPM 的转变开销.

周期任务集产生方法如下:

(1) 任务周期在[1,1000]ms 之间随机产生,服从对数均匀分布(即任务周期的自然对数服从均匀分布).使

用对数均匀分布,能够在短周期(1ms~10ms)、中周期(10ms~100ms)和长周期(100ms~1000ms)等每个时间范围内产生相等数量的任务,这样就模拟了绝大多数硬实时应用;

(2) 任务利用率采用 UUnifast 算法[27]来产生,但丢弃具有任务利用率大于 1 的任务集.使用 UUnifast 算法不仅有助于分析利用率与任务集中任务个数的关系,还可以产生任务利用率的无偏分布;

(3) 任务最坏执行时间直接根据所选择的利用率和周期计算得到 $C_i=U_i\cdot T_i$;

(4) 任务时限在$[C_i,2\cdot T_i]$之间均匀分布.

非周期任务集也可参照上述方式随机产生:任务时限在[1,1000]ms 之间产生,服从对数均匀分布;任务密度采用 UUnifast 算法产生,但丢弃具有任务密度大于 1 的任务集;任务最坏执行时间则根据 $C_i=\lambda_i\cdot D_i$ 直接获得.

每次实验按照上述方法产生任务集,可以保证任务符合一般任务模型.同时,对随机产生的任务集进行可调度性测试,以保证在每种参数设置中至少有 100 组可调度的任务集,取平均能耗值.模拟时间为每组任务集的“超周期”,即所有任务周期的最小公倍数.所有任务属性都在每个超周期内重复出现,从而有助于较为真实地反映算法性能.

4.2 实验结果

为了评价算法的性能,引入如下环境参数:

(1) 任务平均执行时间(average execution time,简称 AET)与最坏情况执行时间(WCET)之比,即

AET/WCET,

刻画了任务实际执行的动态变化情况;

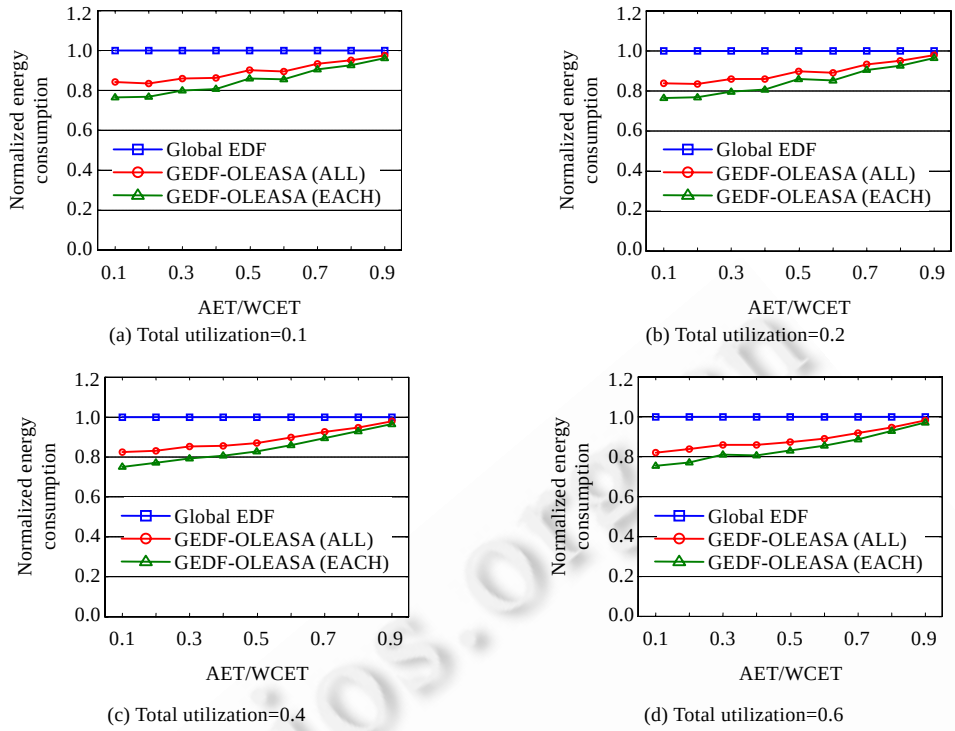
(2) 非周期负载因子(aperiodic load factor),定义为非周期任务的计算需求在整个任务集计算需求中所占比重,刻画了任务集中非周期任务的负载变化情况;

(3) 总利用率(total utilization),定义为 $\sum_{\tau_i\in\Gamma}U_i$ 或 $\sum_{\tau_i\in\Gamma}\lambda_i$, 刻画了系统的实际负载变化情况.

首先研究 AET/WCET 对算法性能的影响.处理器核数为 2,总利用率分别设为 0.1,0.2,0.4 和 0.6,非周期负载因子设为 0.1.AET/WCET 从 0.1 增加到 0.9,步长为 0.1,任务的实际情况执行时间(actual case execution time,简称 ACET)根据 AET/WCET 的值来确定,以 AET 为均值,服从[AET–0.1WCET,AET+0.1WCET]均匀分布.设置 10 个任务组成的任务集,随机产生 100 组可调度的任务集,取平均能耗值.模拟时间为超周期.以 Global EDF 的能耗为标准进行归一化,结果如图 5 所示.当 AET 远低于 WCET 时,GEDF-OLEASA(ALL)比 Global EDF 节能近 18%;而 GEDF-OLEASA(EACH)节能更多,达到 20%以上.随着 AET/WCET 比值的增大,两种算法的节能效果逐渐降低.但是,不论比值如何,两种算法始终比 Global EDF 更节能.

然后研究非周期负载因子对算法性能的影响.处理器核数设为 4,任务集中任务个数分别为 10 和 20,AET/WCET 为 0.3,任务的实际情况执行时间(actual case execution time,简称 ACET)服从[0.2WCET,0.4WCET]均匀分布.总利用率从 0.4 增加到 3.6,步长为 0.4.非周期负载因子从 0.1 增加到 0.9,步长为 0.1.同样,针对每种情况随机产生 100 组可调度的任务集,取平均能耗值.模拟时间为超周期.仍以 Global EDF 的能耗为标准进行归一化,如图 6 所示.对图 6(a)和图 6(b)进行比较可知,随着任务个数的增加,GEDF-OLEASA 算法可利用的动态松弛时间也随之增大,GEDF-OLEASA 算法的节能更多.同时,随着非周期负载因子和总利用率的增大,GEDF-OLEASA(ALL)和 GEDF-OLEASA(EACH)的节能程度逐渐增加.当非周期负载因子和总利用率的数值较大时,前者最多可节能近 8%,而后者在任何参数设置中保持更好的节能效果,最多可节能 12%.

最后分析总利用率对算法性能的影响.由图 5 和图 6 可知,在同样的参数设置下,随着总利用率逐渐增大,GEDF-OLEASA 算法的节能效果变化不大,总利用率对算法性能的影响较小.

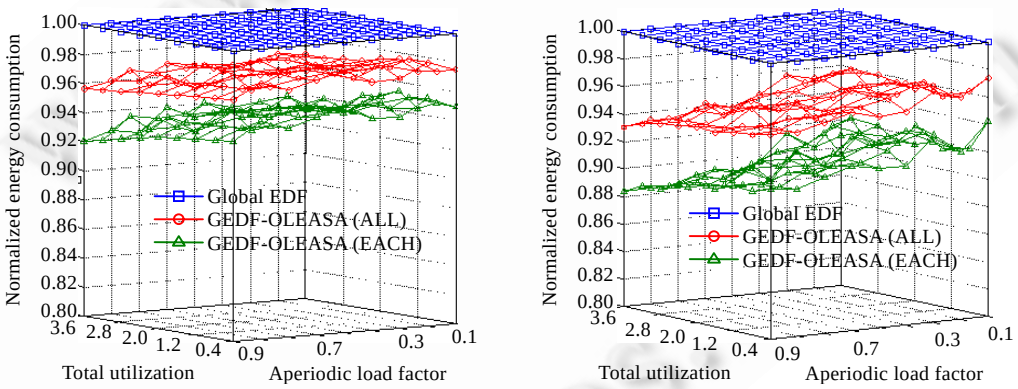


(a) Total utilization=0.1 (b) Total utilization=0.2

(c) Total utilization=0.4 (d) Total utilization=0.6

Fig.5 2 cores, 10 tasks, aperiodic load factor=0.1, total utilization=0.1, 0.2, 0.4 and 0.6

图 5 2 个处理器核,10 个任务,非周期负载因子为 0.1,总利用率为 0.1,0.2,0.4 和 0.6



(a) 4 cores, 10 tasks, AET/WCET=0.3 (b) 4 cores, 20 tasks, AET/WCET=0.3

Fig.6 4 cores, 10 tasks and 20 tasks, AET/WCET=0.3, total utilization ranged from 0.4 to 3.6, aperiodic load factor ranged from 0.1 to 0.9

图 6 4 个处理器核,10 个任务和 20 个任务,AET/WCET 为 0.3,总利用率从 0.4 增加到 3.6,非周期负载因子从 0.1 增加到 0.9

5 结束语

本文基于 Global EDF 调度算法,针对周期和非周期任务组成的一般任务模型,利用 DPM 和片上 DVFS 技术,提出了一种同构多核系统中非常简洁的在线节能调度方法.计算复杂度仅有 $O(\log_2 m)$,且易于在实时操作系

统中实现.该算法能够回收动态松弛时间,降低任务的执行速度,从而减少系统能耗.由于任务的实际执行时间通常远小于最坏情况执行时间,所以 GEDF-OLEASA 算法始终比 Global EDF 算法更节能,最多可节能 15%~20%,最少可节能 5%~10%.

由于事先无法获知每个任务的实际执行情况,所以事实上不可能获得最优的在线节能调度算法.今后,我们将继续考虑进一步提高本文算法的节能效果,在节能效果与计算复杂度之间取得合理折衷.此外,本文将考虑该算法在异构多核系统中的可扩展性问题.

致谢 在此,谨向对本文评审中提出宝贵建议的匿名审稿专家以及对本文的工作给予支持和建议的同行,尤其是德国 KIT 大学 Jian-Jia Chen 副教授、国防科学技术大学郭得科副教授表示诚挚的感谢.

References:

[1] Rele S, Pande S, Onder S, Gupta R. Optimizing static power dissipation by functional units in superscalar processors. Lecture Notes in Computer Science, 2002,2304:85−100. [doi: 10.1007/3-540-45937-5_19]

[2] Chandrakasan AP, Sheng S, Brodersen RW. Low-Power CMOS digital design. IEEE Journal of Solid-State Circuit, 1992,27(4): 473−484. [doi: 10.1109/4.126534]

[3] Kim W, Gupta MS, Wei GY, Brooks D. System level analysis of fast, per-core DVFS using on-chip switching regulators. In: Proc. of the IEEE 14th Int'l Symp. on High Performance Computer Architecture. Los Alamitos: IEEE, 2008. 123−134. [doi: 10.1109/HPCA.2008.4658633]

[4] Pillai P, Shin KG. Real-Time dynamic voltage scaling for low-power embedded operating systems. In: Proc. of the 18th ACM Symp. on Operating Systems Principles. New York: ACM, 2001. 89−102. [doi: 10.1145/502059.502044]

[5] Aydin H, Yang Q. Energy-Aware partitioning for multiprocessor real-time systems. In: Proc. of the 17th Int'l Symp. on Parallel and Distributed Processing. Los Alamitos: IEEE, 2003. 113.2. [doi: 10.1109/IPDPS.2003.1213225]

[6] Chen JJ, Yang CY, Kuo TW. Slack reclamation for real-time task scheduling over dynamic voltage scaling multiprocessors. In: Proc. of the IEEE Int'l Conf. on Sensor Networks, Ubiquitous, and Trustworthy Computing. Los Alamitos: IEEE, 2006. 358−367. [doi: 10.1109/SUTC. 2006.1636201]

[7] Chen JJ, Hsu HR, Kuo TW. Leakage-Aware energy-efficient scheduling of real-time tasks in multiprocessor systems. In: Proc. of the IEEE Real-Time and Embedded Technology and Applications Symp. Los Alamitos: IEEE, 2006. 408−417. [doi: 10.1109/RTAS.2006.25]

[8] Anderson JH, Baruah SK. Energy-Efficient synthesis of periodic task systems upon identical multiprocessor platforms. In: Proc. of the 24th Int'l Conf. on Distributed Computing Systems. Los Alamitos: IEEE, 2004. 428−435. [doi: 10.1109/ICDCS.2004.1281609]

[9] Funaoka K, Kato S, Yamasaki N. Energy-Efficient optimal real-time scheduling on multiprocessors. In: Proc. of the 11th IEEE Symp. on Object Oriented Real-Time Distributed Computing. Los Alamitos: IEEE, 2008. 23−30. [doi: 10.1109/ISORC.2008.19]

[10] Cho H, Ravindran B, Jensen ED. An optimal real-time scheduling algorithm for multiprocessors. In: Proc. of the 27th IEEE Real-Time System Symp. Los Alamitos: IEEE, 2006. 101−110. [doi: 10.1109/RTSS.2006.10]

[11] Nelis V, Goossens J. MORA: An energy-aware slack reclamation scheme for scheduling sporadic real-time tasks upon multiprocessor platforms. In: Proc. of the 15th IEEE Int'l Conf. on Embedded and Real-Time Computing Systems and Applications. Los Alamitos: IEEE, 2009. 210−215. [doi: 10.1109/RTCSA.2009.30]

[12] Yang C, Chen JJ, Luo T. An approximation algorithm for energy-efficient scheduling on a chip multiprocessor. In: Proc. of the Design, Automation and Test in Europe Conf. and Exhibition. Los Alamitos: IEEE, 2005. 468−473. [doi: 10.1109/DATE.2005.51]

[13] Bautista D, Sahuquillo J, Hassan H, Petit S, Duato J. A simple power-aware scheduling for multicore systems when running real-time applications. In: Proc. of the 22nd IEEE/ACM Int'l Parallel and Distributed Processing Symp. ACM/IEEE, 2008. 1−7. [doi: 10.1109/ IPDPS.2008.4536220]

[14] Huang X, Li KL, Li RF. A energy efficient scheduling base on dynamic voltage and frequency scaling for multi-core embedded real-time system. Lecture Notes in Computer Science, 2009,5574:137−145. [doi: 10.1007/978-3-642-03095-6_14]

[15] Rotem E, Mendelson A, Ginosar R, Weiser U. Multiple clock and voltage domains for chip multi processors. In: Proc. of the 42nd Annual IEEE/ACM Int'l Symp. on Microarchitecture. ACM/IEEE, 2009. 459−468. [doi: 10.1145/1669112.1669170]

[16] de Langen P, Juurlink B. Leakage-Aware multiprocessor scheduling. Journal of Signal Processing System, 2009,57(4):73−88. [doi: 10.1007/ s11265-008-0176-8]

[17] Davis RI, Burns A. A survey of hard real-time scheduling algorithms and schedulability analysis techniques for multiprocessor systems. Technical Report, YCS-2009-443, Department of Computer Science, University of York, 2009. https://www.cs.york.ac.uk/ftpdir/reports/2009/YCS/443/YCS-2009-443.pdf

[18] Zhu DK. Reliability-Aware dynamic energy management in dependable embedded real-time systems. In: Proc. of the IEEE Real-Time and Embedded Technology and Applications Symp. Los Alamitos: IEEE, 2006. 397−407. [doi: 10.1109/RTAS.2006.36]

[19] Irani S, Shukla S, Gupta R. Algorithms for power savings. In: Proc. of the 14th Annual ACM-SIAM Symp. on Discrete Algorithms. New York: ACM, 2003. 37−46.

[20] Bansal N, Kimbrel T, Pruhs K. Speed scaling to manage energy and temperature. Journal of the ACM, 2007,54(1):1−39. [doi: 10.1145/1206035.1206038]

[21] Xu RB, Zhu DK, Rusu C, Melhem R, Mossé D. Energy-Efficient policies for embedded clusters. In: Proc. of the ACM SIGPLAN/SIGBED Conf. on Languages, Compilers, and Tools for Embedded Systems. New York: ACM, 2005. 1−10. [doi: 10.1145/1065910.1065912]

[22] Xu RB, Mossé D, Melhem RG. Minimizing expected energy in real-time embedded systems. In: Proc. of the 5th ACM Int'l Conf. on Embedded Software. New York: ACM, 2005. 251−254. [doi: 10.1145/1086228.1086274]

[23] ARM CORPORATION. 2010. http://www.arm.com/products/processors/classic/arm11/arm11-mpcore.php

[24] Dorsey J, Searles S, Ciraula M, Johnson S, Bujanos N, Wu D, Braganza M, Meyers S, Fang E, Kumar R. An integrated quad-core opteron processor. In: Proc. of the IEEE Int'l Solid State Circuits Conf. Los Alamitos: IEEE, 2007. 102−103. [doi: 10.1109/ISSCC.2007.373608]

[25] Krishna CM. Shin KG. Real-Time Systems. New York: McGraw-Hill, 1997. 1−9.

[26] Baker TP. Multiprocessor EDF and deadline monotonic schedulability analysis. In: Proc. of the 24th IEEE Real-Time Systems Symp. Los Alamitos: IEEE, 2003. 120−129. [doi: 10.1109/REAL.2003.1253260]

[27] Bini E, Buttazzo GC. Measuring the performance of schedulability tests. Journal of Real-Time Systems, 2005,30(1-2):129−154. [doi: 10.1007/s11241-005-0507-9]

张冬松(1980－),男,河南信阳人,博士生,CCF 学生会员,主要研究领域为实时系统.

陈芳园(1982－),女,博士生,主要研究领域为计算机体系结构,嵌入式实时.

吴彤(1979－),男,博士,讲师,主要研究领域为实时系统,军事高科技.

金士尧(1937－),男,教授,博士生导师,CCF 高级会员,主要研究领域为实时系统,分布式仿真,虚拟现实.