软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(4):819−834 [doi: 10.13328/j.cnki.jos.004752]

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

一种能量收集嵌入式系统自适应调度算法*

葛永琪[1], 董云卫[1], 张 健[1], 顾 斌[1,2]

[1](西北工业大学 计算机学院,陕西 西安 710129)

[2](北京控制工程研究所,北京 100190)

通讯作者: 葛永琪, E-mail: geyongqi@mail.nwpu.edu.cn

摘 要: 能量收集嵌入式系统(energy harvesting embedded system,简称 EHES)的任务调度算法需要考虑能量收集单元的能量输出、能量存储单元的能量水平和能量消耗单元的能耗.实时任务在满足能量约束的条件下,才可能满足时间约束.在这个背景下,传统固定优先级调度算法不再适用于 EHES.提出一种基于分组的自适应任务调度算法,它能根据能量收集单元由于能量输出的不确定性而造成的非能量约束情况和能量约束情况,自适应地选择任务调度算法.在非能量约束的情况下,减少任务抢占次数,增强任务的可调度性;在能量约束情况下,减少电池模式切换次数,提高能量存储单元的平均能量水平,从而降低系统能量约束.在一个可进行大范围任务集合仿真的实验环境下对提出的算法进行验证,并将基于分组的自适应调度算法与现有的两个经典算法进行了对比.

关键词: 能量收集;实时性;嵌入式系统;信息物理融合系统;自适应调度

中图法分类号: TP316

中文引用格式: 葛永琪,董云卫,张健,顾斌.一种能量收集嵌入式系统自适应调度算法.软件学报,2015,26(4):819−834. http://www.jos.org.cn/1000-9825/4752.htm

英文引用格式: Ge YQ, Dong YW, Zhang J, Gu B. Adaptive scheduling algorithm for energy harvesting embedded system. Ruan Jian Xue Bao/Journal of Software, 2015,26(4):819−834 (in Chinese). http://www.jos.org.cn/1000-9825/4752.htm

Adaptive Scheduling Algorithm for Energy Harvesting Embedded System

GE Yong-Qi[1], DONG Yun-Wei[1], ZHANG Jian[1], GU Bin[1,2]

[1](School of Computer Science, Northwestern Polytechnical University, Xi'an 710129, China)

[2](Beijing Institute of Control Engineering, Beijing 100190, China)

Abstract: The task scheduling of energy harvesting embedded systems (EHES) should take into account the energy supply of energy harvesting unit, the energy level of energy storage unit and the energy consumption of energy dissipation unit. A real-time task can meet time constraint only if its energy constraint is satisfied. Against this background, conventional fixed-priority tasks scheduling algorithms are not suitable for EHES. A group-based adaptive task scheduling algorithm is proposed in this paper. It can select suitable task scheduling algorithm adaptively according to the non-energy constraint condition and the energy constraint condition caused by the uncertain energy supply of energy harvesting unit. In the case of non-energy constraints, the algorithm can reduce the tasks preemptions and enhance the tasks schedulability. In the case of energy constraints, the algorithm can reduce the battery-mode switches and increase the average energy level of energy storage unit, thus decrease the system energy constraint. The proposed algorithm is validated with large scale simulations comparing with other two existing classical algorithms.

Key words: energy harvesting; real-time; embedded system; cyber-physical system; adaptive scheduling

近年来,随着嵌入式技术和计算机技术的发展,人们对嵌入式智能设备的需求不再仅仅局限于实时性以及功能扩充,而更关注嵌入式计算和物理环境的紧密耦合.很多嵌入式系统是由电池供电的,对设备能量的随意使

* 基金项目: 国家高技术研究发展计划(863)(2011AA010105)
收稿时间: 2014-07-01; 修改时间: 2014-10-14; 定稿时间: 2014-11-14

用会缩短设备的运行时间.动态调压调频技术(dynamic voltage and frequency scaling,简称 DVFS)[1−3]和动态电源管理(dynamic power management,简称 DPM)[4−6]可以在满足时间约束的条件下有效降低系统能耗,但在实际应用中,有些设备被部署后,相应的嵌入式应用需要运行很长的时间,电池总是会被耗尽,并且更换电池又不可行,采用能量收集技术[7,8]从周围物理环境资源(例如阳光、风、振动和潮汐等)收集能量,是解决这个问题的一个有效方法.因此,能量收集技术可以被用来为嵌入式设备供电和为电池补充能量,延长设备的工作寿命.我们将采用能量收集技术供电的嵌入式系统称为能量收集嵌入式系统(energy harvesting embedded system,简称 EHES).EHES 的能量收集和存储都需要时间,能量收集所需时间会导致任务调度过程中产生空白时间,所以,EHES 的调度器不是连续工作的.传统调度算法(例如截止时间优先算法、速率单调算法等)没有考虑物理环境的影响(能量收集单元能量输出的不确定性),会造成任务由于能量不足而错过截止时间,因此,传统调度算法不再适用于 EHES.Allavena 和 Mossé[9]首先提出采用能量收集技术供电的嵌入式系统任务调度问题,接着,一些解决此任务调度问题的算法被提了出来.LSA 算法[10]可以根据任务能耗调节 CPU 频率以调整最坏执行时间(worst case execution time,简称 WCET),研究结果依赖于很强的假设,任务能耗直接关联于 WCET,这种假设不切合实际[11].Abdeddaïm 等人提出了 ALAP 算法[12]和 ASAP 算法[13],并证明了在同时考虑能量和时间限制的条件下, ASAP 算法是最优的[14],并对基于固定优先级的实时能量收集系统的可调度性进行了分析[15].我们可以将目前的算法划分为两类:能量贪婪算法和计算贪婪算法.能量贪婪算法通过延迟任务执行提供尽可能多的松弛时间来为能量存储单元补充能量;计算贪婪算法优先执行计算任务,只有当能量不足时才给能量存储单元补充能量. ALAP 算法和 ASAP 算法分别是能量贪婪算法和计算贪婪算法的代表.为了获得更好的系统性能,并同时对系统能耗进行优化,一些方法将任务调度与 DVS 技术相结合进行研究.EA-DVFS 算法[16]判断当前任务:如果没有足够的能量运行任务,就降低 CPU 频率进行节能;否则,CPU 以最大频率运行任务.HA-DVFS 算法[17]在 EA-DVFS 算法的基础上,更进一步地优化系统性能和提高能量利用率.它在过流的情况下,利用过流的能量提升 CPU 频率,贡献更多的松弛时间来降低后续任务的执行频率.但在一些高可靠实时嵌入式系统中,DVFS 方法会延长任务的执行时间,影响系统的实时性.因此,DVFS 方法会影响这类系统的可靠性.

由于能量收集单元能量输出的不确定性,造成 EHES 任务调度存在非能量约束和能量约束两种情况.以月球车为例,由于太阳电池阵受所在物理环境(例如经度、纬度、天气、温度等)的影响,其能量供给不稳定,而且不能存储能量,因此,月球车系统配备电池来存储和调节电能[18].当太阳电池阵的能量输出大于负载需求时,负载由太阳电池阵供电,并且电池能量处于满荷状态,电池不影响任务调度,这属于非能量约束情况.由于电池存储能量的限制,非能量约束情况下会造成能量浪费,这种情况下应追求最大系统性能,传统调度算法适用此种情况;当太阳电池阵的能量输出大于负载需求时,多余能量为电池充电,或者当太阳电池阵的能量输出小于负载需求,需要电池放电来维持能量供给时,需要电池参与任务调度,任务调度需要考虑任务能耗和电池能量的限制,这属于能量约束情况.然而,目前非能量约束情况下调度算法不适用于能量约束情况,因为其没有考虑电池能量的变化,会造成任务由于能量不足而错过截止时间.目前能量约束情况下的调度算法可应用于非能量约束情况,但却没有对系统性能进行优化.因此,本文研究的挑战在于以下两点:(1) 如何从能量角度,将 EHES 任务调度问题化解为非能量约束和能量约束下的两个子问题;(2) 如何使得 EHES 任务调度具备一定的自适应性,可以适应非能量约束情况和能量约束情况下对性能和能量的不同需求.

在 EHES 任务调度过程中,能量输出单元与能量存储单元共同形成物理环境,其能量变化是连续的物理过程,物理过程为任务执行计算环境提供重要的计算信息,计算环境的智能决策反过来影响物理环境.EHES 通过计算和物理持续交互和融合来进行系统资源的有效分配和系统性能优化.因此,EHES 是一种信息物理融合系统(cyber-physical system,简称 CPS)[19].本文在 CPS 背景下,结合月球车应用,通过研究计算和物理的交互过程,基于固定优先级抢占式调度,提出一种基于分组的自适应任务调度算法(group-based adaptive task scheduling algorithm,简称 GATS).该算法通过能量约束判断条件自适应地选择不同的调度算法,在非能量约束情况下,通过减少任务抢占,优化系统性能;对于能量约束情况,在满足实时性的前提下,减少电池模式切换次数,提高能量存储单元平均能量水平,从而降低能量约束.

1 系统模型

CPS 是计算进程和物理进程的紧密耦合,通过计算进程与物理进程的持续交互,实现计算进程利用物理进程完成智能决策,而物理进程又借助计算进程实现对物理环境的感知和控制.我们用 C 表示计算单元属性集合,这些属性是应用程序计算函数的输入.P 是物理单元属性集合,包括时变物理量、环境信号等.计算单元属性 C 通过物理进程与物理单元属性 P 关联来改变物理环境.这样的物理进程可以用交互参数 I 表示,交互参数可以连接计算单元属性和物理单元属性.计算单元属性是时间相关的,因此,集合 C 到 I 的映射可以表示为 $G:C\times t\rightarrow I,t$ 表示时间.物理单元属性通常是时空相关的,因此物理属性到交互参数的映射可以表示为 $H:P\times t\times\{x,y,z\}\rightarrow I$,$\{x,y,z\}$表示空间坐标的一个坐标点.在实际应用中,映射 G 可以通过实验或者执行算法而获得,映射 H 可通过建立物理进程模型获得.下面给出信息物理交互的定义[20].

定义 1. 一个信息物理交互 K 是从 I 的子集到 P 或 C 子集的逆映射.

以月球车系统为例,给出了 EHES 自适应任务调度的抽象结构,如图 1 所示.在月球车任务调度过程中,能量存储单元的当前能量(P)和能量收集单元的能量输出(P)会通过映射 H(可以通过建立物理模型获得)影响当前系统交互参数可用能量(I),调度器通过感知可用能量和任务能量属性,自适应地调整任务调度策略,使得系统运行在最优的工作状态,并适应能量收集单元能量输出的不确定性.这是从 I 到 P 的逆映射 K.同时,任务的时间属性、能量属性和优先级属性(C)会由于自适应任务调度器选择不同的任务调度算法而影响交互参数空闲时间(I),空闲时间由任务调度算法(G)决定.能量存储单元利用空闲时间补充能量,进而影响能量存储单元的充放电模式(P).这是从 I 到 C 的逆映射 K.因此,可用能量作为任务调度算法的参数,影响任务调度的实时性.同时,任务调度算法在满足时间约束的情况下,结合任务能量属性,提供空闲时间为能量存储单元补充能量,使能量存储单元维持较高的能量水平,降低系统能量约束,形成满足系统实时性能和降低系统能量约束力的控制过程.

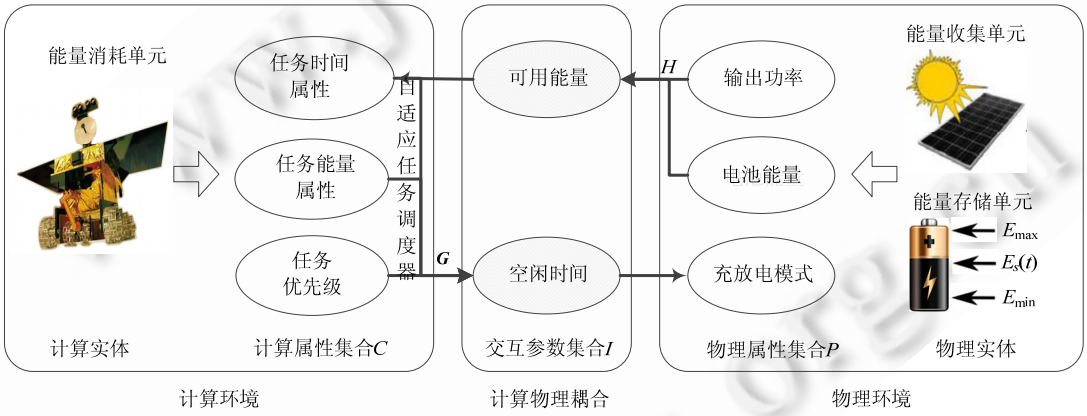


Fig.1 Abstraction of EHES adaptive task scheduling

图 1 EHES 自适应任务调度的抽象结构

EHES 由 3 部分组成:能量收集单元、能量存储单元和能量消耗单元.能量收集单元从周围环境收集能量,当能量收集单元能量输出不足时,能量存储单元释放能量来维持任务执行;能量存储单元通常是可充电电池或超级电容;能量消耗单元是指运行实时任务的嵌入式系统,它的能量来源于能量收集单元和能量存储单元.当能量收集单元的能量输出超出能量消耗单元需求时,多余的能量给能量存储单元补充能量;当能量收集单元的能量输出不能满足能量消耗单元需求时,不足的能量由能量存储单元放电进行补充.基于文献[21]的系统模型,下面给出能量收集单元模型、能量存储单元模型和能量消耗单元模型,我们用 H 表示能量收集单元,S 表示能量存储单元,W 表示能量消耗单元.

1.1 能量收集单元模型

由于受环境影响,能量收集单元的能量输出不是恒定的,是随时间变化的[21].能量收集单元一方面为能量消

耗单元提供能量,另一方面可以为能量存储单元补充能量.能量收集单元可以定义其功率输出为时间相关函数 $P_h(t)$,在时间间隔$[t_1,t_2]$的能量输出 $E_h(t_1,t_2)$如公式(1)所示:

$$E_h(t_1,t_2)=\int_{t_1}^{t_2}P_h(t)\mathrm{d}t \tag{1}$$

我们假设 $P_h(t)$是一个常数,即 $P_h(t)=P_h$,在时间间隔$[t_1,t_2]$的能量输出用公式(2)表示:

$$E_h(t_1,t_2)=(t_2-t_1)\times P_h \tag{2}$$

1.2 能量存储单元模型

能量存储单元的最大容量用 F 表示,能量补充速率用 e_{bat} 表示,即使在任务执行期间,能量存储单元也可补充能量.电池能量在两个阈值 $E_{\min}$ 和 $E_{\max}$ 间浮动,$E_{\max}$ 是能量存储单元充电允许容量最大值,$E_{\min}$ 是保持系统运行的最小能量需求.能量存储单元在 t 时刻的能量水平记为 $E_s(t)$,则在任意时刻 t,有 $F\geqslant E_{\max}\geqslant E_s(t)\geqslant E_{\min}\geqslant 0$.在时间间隔$[t_1,t_2]$,能量存储单元的能量输出 $E_s(t_1,t_2)$如公式(3)所示:

$$E_s(t_1,t_2)=E_s(t_2)-E_s(t_1),\forall t_1<t_2 \tag{3}$$

可以看出,$E_s(t_1,t_2)$可正可负:当 $E_s(t_1,t_2)$为正,表示在时间间隔$[t_1,t_2]$,能量存储单元处于充电模式;当 $E_s(t_1,t_2)$为负,表示在时间间隔$[t_1,t_2]$,能量存储单元处于放电模式.能量存储单元通常为电池或电容,本文后面提到的能量存储单元以电池代替.

1.3 能量消耗单元模型

能量消耗单元是指由 n 个独立任务构成的集合,表示为 $W=\{\tau_i,i=1,...,n\}$.一个实时任务为五元组 $\tau_i=(C_i,E_i,D_i,T_i,P_i)$,其中,每个任务需要执行 WCET 个时间单位,用 C_i 表示,且需要消耗最坏能量消耗(worst case energy consumption,简称 WCEC)个单位的能量,用 E_i 表示.在时间间隔$[t_1,t_2]$,任务能耗用 $E_w(t_1,t_2)$表示.所有任务能量的消耗是线性的,每个执行时间单位消耗常数单位的能量.任务 τ_i 是强时间约束的,必须在相对截止时间 D_i 之前完成.一个任务 τ_i 释放无限个用 T_i 分割的实例,满足 $0<C_i\leqslant D_i\leqslant T_i$.任务集合按照优先级 P_i 排序,τ_1 是最高优先级任务,τ_n 是最低优先级任务.我们约定:值越小,优先级越高.任务能耗由能量存储单元和收集能量单元联合提供,在时间间隔$[t_1,t_2]$,如果任务可以调度,则任务能耗 $E_w(t_1,t_2)$需满足公式(4):

$$E_w(t_1,t_2)\leqslant E_s(t_1)+E_h(t_1,t_2),\forall t_1<t_2 \tag{4}$$

2 EHES 自适应调度算法

当能量收集单元的能量输出大于能量消耗单元的能量需求,不需要电池释放能量,而且电池能量已达到最大容量,不需要充电,此时,任务运行不需要考虑电池的能量水平,任务调度不受能量约束,调度的目标是提高系统性能;当能量收集单元的能量输出不能满足能量消耗单元的能量需求,需要电池释放能量来满足需求时,任务调度需要先满足能量约束,才能满足时间约束,任务调度的目标降低系统的能量约束,让系统避免由于能量不足而错过截止时间.我们从能量角度,通过建立能量约束判断条件,将 EHES 任务调度划分为两个子问题.在时间间隔$[t_1,t_2]$,能量约束判断条件如公式(5)所示.当满足公式(5)时,表示不受能量约束;反之,表示受能量约束.EHES 自适应调度算法依据此能量约束判断条件,选择不同的任务调度算法.

$$\begin{cases}E_h(t_1,t_2)\geqslant E_w(t_1,t_2), & \forall t_1<t_2\\ E_s(t)=E_{\max}, & \forall t_1<t<t_2\end{cases} \tag{5}$$

2.1 非能量约束情况

在非能量约束的情况下,任务调度的目标是追求最大系统性能.但是,研究者们往往忽略了影响系统性能和能耗的一个重要影响因素——上下文切换开销.随着嵌入式系统对能耗的限制愈加严格,上下文切换的开销已经难以忽略.上下文切换是多进程共享处理器的一种基本机制,包括直接开销和间接开销:直接开销包括保存和恢复 CPU 寄存器、清理 CPU 流水线和执行 OS 调度;间接开销包括进程切换虚拟地址转换引起的 TLB 扰乱.研究表明,间接开销远大于直接开销[22].因此,上下文切换对降低系统开销有着重要的意义.固定优先级抢占阈

值调度(fixed-priority tasks with preemption thresholds scheduling,简称 FPPT)[23]基于单 CPU 固定优先级,每个任务 τ_i 分配一个固定优先级 $\pi_i \in [1,2,\ldots,n]$和一个抢占阈值 $\gamma_i \in [1,\pi_i]$,优先级 π_i 和强制阈值 γ_i 都是固定的.假设任务之间是独立的(没有因为共享资源而引起阻塞),除非任务完成计算,否则任务不会自己挂起.当一个任务被释放时,使用它的固定优先级竞争 CPU.当任务 τ_i 开始执行时,它可以被任务 τ_j 抢占,当且仅当 $\pi_j<\gamma_i$.抢占调度和非抢占调度都是抢占阈值调度的特例.每个任务的抢占阈值等于系统中任务的最高优先级,那么这个调度就是非抢占式的;如果每个任务的抢占阈值与其优先级相等,那么这个调度就是抢占式的.所以,FPPT 具有抢占式和非抢占式两种调度的优点,从而可以使得在一个抢占式或者非抢占式的调度中,不可调度的任务集合变得可以调度,而且可以有效减少抢占次数.

基于 FPPT 调度策略,本文给出了抢占阈值调度实现算法,通过设定防止抢占判断条件,避免抢占发生.抢占阈值的分配直接与最坏响应时间(worst-case response time,简称 WCRT)[23]相关,然而,FPPT 调度策略的 WCRT 计算模型忽略了上下文切换开销,会影响 WCRT 计算的准确性,造成任务执行的延迟,进而会影响整个系统的可调度性.

本文在考虑上下文切换开销的基础上对 WCRT 计算模型进行了扩展.下面先给出抢占阈值调度实现算法.

2.1.1 抢占阈值调度实现算法

当任务分配了抢占阈值,任务就拥有了防止被抢占的保护空间 $[\gamma_i,\pi_i]$,可以防止优先级为 π_j,满足 $\pi_i>\pi_j\geqslant\gamma_i$ 条件的任务 τ_j 的抢占,从而减少上下文切换.算法 1 给出抢占阈值调度实现算法(preemption thresholds scheduling implementation algorithm,简称 PTSI).

算法 1. 抢占阈值调度实现算法(PTSI).

```
1.  //prev 表示当前任务,next 表示从就绪队列选取的下一个任务.
2.  t←0
3.  T_prev←0
4.  P_next←0
5.  loop
6.     W←{τ1,τ2,…,τn}                                  //就绪队列按固定优先级排序
7.     If W≠∅ then
8.        next←τ1                                      //取任务集合 W 中最高优先级任务
9.        T_prev←Current task threshold
10.       P_next←Next task priority
11.       If (P_next≥T_prev) and (prev!=next) and (!lastExcutedJobHasCompleted)   //防止抢占判断条件
12.          next←prev
13.       Endif
14.       执行 next 一个时间单位
15.    Endif
16. t←t+1
17. Endloop
```

PTSI 算法首先从按照固定优先级从高到低排序的任务队列中选取优先级最高的任务,即切换任务 *next*,然后判断 *next* 是否可以抢占当前任务 *prev*,当满足防止抢占判断条件时,则不允许抢占,不进行上下文切换.算法核心是防止抢占条件的判断.由于任务可能主动释放 CPU,也可能由于被抢占而被动释放 CPU,因此任务调度有主动和被动调度之分:对于主动调度,调度器允许任务主动放弃 CPU;对于被动调度,调度器需要比较被抢占任务的抢占阈值和抢占任务优先级之间的大小关系,从而判断是否需要抢占.防止抢占条件需要考虑以下 3 种情况:

1) 当切换任务 *next* 的固定优先级低于当前任务 *prev* 的抢占阈值时,不能发生抢占,即 $P_{next}\geqslant T_{prev}$;

2) 抢占阈值调度不能发生在切换任务 *next* 和当前任务 *prev* 是同一个任务的情况下,此时不发生抢占,即:

prev!=*next*;

3) 抢占阈值调度主要用来减少被抢占次数,因此,防止抢占条件需要屏蔽主动调度情况.主动调度发生在任务执行完成时,主动放弃 CPU,即!*lastExcutedJobHasCompleted*.

2.1.2 可调度性分析

由于任务上下文切换时间开销的累积效应,上下文切换的时间开销会对任务的可调度性带来不可忽视的影响,因此有必要将任务上下文切换的时间开销加入任务最坏响应时间的分析中去.上下文切换分为自愿上下文切换和非自愿上下文切换:自愿上下文切换是指进程由于系统调用返回、调用调度器函数或者显示调用退出函数主动放弃 CPU 而发生的上下文切换,它可以发生在内核态和用户态;非自愿上下文切换发生在进程被抢占而被迫放弃 CPU.一个任务的响应时间由 3 部分组成:1) 任务自身的计算时间,此时应考虑自愿上下文切换开销,任务每主动放弃 CPU 一次,发生自愿上下文切换一次;2) 来自其他高优先级或者相同优先级任务的抢占,此时应考虑非自愿上下文切换开销,任务每被抢占一次,会先切换执行抢占任务,然后在切换回来继续执行当前任务,共发生两次上下文切换;3) 由于抢占阈值导致的来自更低优先级的任务阻塞时间.

如果更低优先级任务正在执行,目标任务不能抢占(由于更高的抢占阈值),这就产生了阻塞时间 $B(\tau_i)$.任务 τ_i 的最大阻塞时间如公式(6)所示,C_j 是阻塞任务的执行时间.

$$B(\tau_i)=\max_j\{C_j :: \gamma_j \leqslant \pi_i < \pi_j\} \tag{6}$$

在最坏情况下,任务刚好在临界时刻之前开始执行.在开始时刻之前到达的所有更高优先级任务和任务 τ_i 的任何更早实例,都应当在这个时刻之前完成.因此,任务 τ_i 的第 q 个实例的开始时间可以通过下面的公式(7)经迭代计算得到,其中,$S_i(q)$表示任务 τ_i 的第 q 个实例的开始时间,C_j 是抢占任务的执行时间,C_{vcsw} 是自愿上下文切换时间,C_{nvcsw} 是非自愿上下文切换时间.

$$S_i(q)=B(\tau_i)+(q-1)\cdot(C_i+C_{vcsw})+\sum_{\forall j,\pi_j<\pi_i}\left(1+\left\lfloor\frac{S_i(q)}{T_j}\right\rfloor\right)\cdot(C_j+2C_{nvcsw}) \tag{7}$$

在任务 τ_i 开始之后的执行期间,只有优先级高于 γ_i 的任务在任务 τ_i 完成之前能够获得处理器资源,抢占仅来自于优先级高于 γ_i 的任务.基于此,计算最坏结束时间的计算公式如公式(8)所示,其中,$F_i(q)$表示任务 τ_i 的第 q 个实例的结束时间.

$$F_i(q)=S_i(q)+(C_i+C_{vcsw})+\sum_{\forall j,\pi_j<\gamma_i}\left(\left\lceil\frac{F_i(q)}{T_j}\right\rceil-\left(1+\left\lfloor\frac{S_i(q)}{T_j}\right\rfloor\right)\right)\cdot(C_j+2C_{nvcsw}) \tag{8}$$

任务 τ_i 的最坏响应时间 R_i 等于任务 τ_i 在忙周期时间间隔内所有实例的最坏响应时间的最大值.R_i 的计算如公式(9)所示,m 表示忙周期的最后一个实例:

$$R_i=\max_{q\in\{1,\ldots,m\}}(F_i(q)-(q-1)\cdot T_i) \tag{9}$$

由公式(7)、公式(8)可知:等式两边均有变量 $S_i(q)$和 $F_i(q)$,直接通过方程式无法计算,需采用迭代算法多次计算 $S_i(q)$和 $F_i(q)$,直到 $S_i(q)$和 $F_i(q)$收敛为止.当任务 τ_i 最坏响应时间 R_i 小于截止时间 D_i 时,任务 τ_i 可调度,如果任务集合中所有任务可调度,则任务集合可调度.

阈值的确定直接影响任务上下文的切换频率.Wang 和 Saksena[23]提出了一种在固定优先级的条件下,系统地分配任务抢占阈值的算法,使得阈值分配能够保证可调度性.该算法建议抢占阈值的分配从最低优先级任务开始到最高优先级任务,因为可调度性分析依赖于比当前任务优先级更高的抢占阈值.在为特定的任务查找最优的抢占阈值时,先从自身的优先级开始一直到系统的最高优先级,遇到第 1 个使得任务可调度的抢占阈值就停止.本文采用该算法对实时任务的抢占阈值进行分配,若抢占阈值分配失败,则调整任务集合属性.上下文切换开销可以通过在上下文切换函数前后插入时间函数获取,多个任务取上下文切换时间的最大值.

2.2 能量约束情况

在能量约束情况下,传统调度算法不再适用能量收集系统.调度算法需要首先保证能量约束,才能保证时间

约束.因此,能量约束情况下,应尽可能地降低能量约束.要降低能量约束,应使电池更长时间地处于高能量水平,拥有更多的能量来供能量消耗单元使用.以磷酸铁锂电池为例:容量衰退至 85%之前,深充深放与浅充浅放的使用模式对于电池能量转移能力的影响几乎是相同的;当电池容量衰退至 75%时,深充深放的使用模式在电池能量转移总量和能量效率上均优于浅充浅放的使用模式[24].因此在任务负载较高时,采用深充深放模式,从而减少了电池浅充浅放模式的使用次数,可以提高电池的能量转移总量和能量使用效率.而且,目前的调度算法没有考虑过于频繁的电池模式切换带来的开销,频繁的电池模式切换使得电池过多地处于浅充浅放模式,降低了能量转移能力,在实际应用中,这是不可取的[13].如果在满足时间约束的情况下,让任务集中到一起运行,此时电池处于放电模式,在任务松弛时间为电池补充能量,此时电池处于充电模式.这样就可以减少电池模式切换,提高电池的能量转移效率,使得电池处于高能量水平,进而可降低系统能量约束.

2.2.1 减少电池模式切换的调度算法

下面,本文将提出一种减少电池模式切换的调度算法(battery-mode reduction task scheduling algorithm,简称BSRTS).就绪队列任务按照优先级由高到低排序,在满足任务截止时间约束和能量约束的条件下,尽量让电池保持充电或者放电模式.假设当前电池处于放电模式,就绪队列有任务要执行,如果电池能量满足任务运行,即使该任务之前有空闲时间,那么该任务也立刻执行;如果当前电池在利用松弛时间[25]充电,有任务来临且有足够的能量来执行任务,则判断任务消耗速率是否大于能量补充速率,即 $E_i/C_i>P_h$,并判断任务是否有松弛时间.如果满足这两个条件,则充电松弛时间个单位.这样,任务集中连续执行,电池保持放电模式,松弛时间集中到一起,电池保持充电模式,从而减少了电池模式的切换,保持了电池深充深放的使用模式.

算法 2. 减少电池模式切换调度算法(BSRTS).

```
1.  //prev 表示当前任务,next 表示从就绪队列选取的切换任务.
2.  t←0
3.  chflag←0                          //电池模式标示,0 表示放电,1 表示充电
4.  st←0                              //松弛时间
5.  loop
6.     W←{τ1,τ2,…,τn}                 //就绪队列按固定优先级排序
7.     If W≠∅ then
8.        next←τ1                     //取任务集合 W 中最高优先级任务
9.        If chflag==0
10.          If Ph(t)+Es(t)−Emin≥Ek/Ck then
11.             执行 next 一个时间单位
12.          Else
13.             chflag←1
14.          Endif
15.       Else
16.          st←getslacktime          //获得松弛时间
17.          If st>0 && Ei/Ci>Ph
18.             充电 st 个时间单位
19.          Endif
20.       Endif
21.    Endif
22.    t←t+1
23. Endloop
```

BSRTS 算法使用 *chflag* 来标记电池模式时,当电池处于放电模式,使用 ASAP 算法思想,判断能量是否满足

任务执行一个时间单位,若满足,则立刻执行一个时间单位.当没有任务执行时,转为充电模式,充电模式发生在两种情形下:1) 任务存在松弛时间;2) 任务的能量消耗速率小于能量补充速率.

2.2.2 可调度性分析

在能量约束情况下,任务调度需要同时满足时间约束和能量约束.我们需要分析最坏情况下处理器需求时间和最坏情况下补充任务能量需求所需要的时间.在最坏情况下,电池处于最低能量水平,所有任务同时释放.由于BSRTS算法基于ASAP算法思想,因此,我们可依据ASAP算法的最坏响应时间计算方法[14]分析可调度性.在能量约束的情况下,基于固定优先级的最坏响应时间分析方法需要在最坏响应时间计算过程中考虑任务能量消耗.通过逐步计算优先级为 i 的任务的响应时间.最坏响应时间是同时满足最坏处理器需求时间和能量需求的最大时间需求.

在 t 时刻,优先级为 i 的任务的最坏处理器需求时间 $WP_i(t)$为在时间间隔$[0,t]$,执行优先级为 $1,...,i-1,i$ 的任务执行时间与松弛时间 $S_i(t)$总和,可以使用公式(10)计算:

$$WP_i(t)=\sum_{j\leqslant i}\left\lceil\frac{t}{T_j}\right\rceil\times C_j+\min_{j\leqslant i}S_j(t) \tag{10}$$

在 t 时刻,优先级为 i 的任务的最坏情况能量需求为在时间间隔$[0,t]$,执行高于和等于 i 的任务,即,优先级为 $1,...,i-1,i$ 的任务需要补充的能量总和,可以使用公式(11)计算.由于电池有初始能量,因此实际需要补充能量需要减去初始能量 $E(0)$.

$$WE_i(t)=\sum_{j\leqslant i}\left\lceil\frac{t}{T_j}\right\rceil\times E_j-E(0) \tag{11}$$

优先级为 i 的任务在 t 时刻的最坏响应时间 $W_i(t)$为在时间间隔$[0,t]$,同时满足最坏处理器需求和最坏能量需求的最大时间,用公式(12)计算获得.如果最坏响应时间小于截止时间,则该任务可调度;如果任务集合中所有任务满足该条件,则任务集可调度.

$$W_i(t)=\max\left(\left\lceil\frac{WE_i(t)}{P_h}\right\rceil,WP_i(t)\right) \tag{12}$$

2.3 基于分组的自适应任务调度算法

非能量约束的情况下,由于抢占阈值的引入,PTSI 算法会影响系统中其他任务的正常运行.如果将 PTSI 算法也应用到能量约束的情况下,会使得优先级低的任务由于避免抢占而消耗了应该分配给高优先级任务执行的能量,从而导致高优先级任务由于能量不足而错过截止时间.而且,在 EHES 中存在多种类型的任务,有些任务为操作系统自身功能服务,有些则为应用任务.不同类型的任务,调度策略也各有不同.通过对任务进行分组,能够隔离不同类型的任务,避免作用于某些类型任务之上的调度策略对其他类型的任务产生影响,进而避免了对操作系统调度系统造成紊乱,影响操作系统的可靠性.

因此,本文给出基于分组的自适应调度算法(group-based adaptive task scheduling algorithm,简称 GATS)来对系统任务进行隔离,并通过能量约束条件判断自适应地选择不同的调度算法.

2.3.1 基于分组的自适应调度算法

在对任务分组之前,任务按照优先级由高到低排序.为避免系统关键任务和其他任务之间互相影响,将任务分为系统任务和应用任务:系统任务是为操作系统提供系统功能服务,运行于操作系统中的服务程序,如时钟中断、调度程序、驱动程序等,其优先级区间为$[0,low)$;应用任务是指运行于操作系统之上,提供特定功能的应用程序,其优先级区间为$[low,high]$.我们使用任务标记 $Tflag$ 区别系统任务和应用任务,$Tflag$=0 表示为系统任务,$Tflag$=1 表示为应用任务.我们将系统任务分为一组,使用系统默认调度策略.根据基于 L_i 级应用任务的定义将应用任务分为若干组.下面给出 L_i 级应用任务的定义.

定义 2. 定义任务优先级子区间 $L_i\subseteq[low,high]$,L_i 不能重叠,如果任务 τ_i 的优先级 $\pi_i\in L_i$,则任务 τ_i 为 L_i 级应用任务.

对任务进行分类和标记后,GATS 算法的分组规则如下:

规则 1. 系统任务分组规则:如果任务 τ_i 的优先级 $\pi_i \in [0,low)$,并且 $Tflag=0$,则该任务为系统任务,加入系统任务组,记为 G_0.

规则 2. 应用任务分组规则:如果任务 τ_i 的优先级 $\pi_i \in L_i$,并且 $Tflag=1$,则该任务为应用任务,加入应用任务组,记为 G_i.

如果任务优先级范围为[0,*high*],系统任务优先级区间为[0,*low*),应用任务优先级区间为[*low*,*high*].根据对操作系统中任务的分类,将任务划分为系统任务和应用任务.根据分组规则,所有系统任务分为一组,即,系统任务分组数为 1.应用任务优先级区间为[*low*,*high*],在极端情况下,如果 L_i=[*low*,*high*],即,将所有应用任务分为一组,则应用任务最少可以分为 1 组;如果将[*low*,*high*]中每一级优先级对应的应用任务各分为一组,则自定义任务最多可以分为 *high*–*low*+1 组.整个系统的任务分组数等于自定义任务的分组数与系统任务分组数之和,即,系统总分组数处于[2,*hight*–*low*+2]之间.

假设操作系统中共有 n 个实时任务 $\tau_1,\tau_2,\ldots,\tau_n$,分成两个任务组 G_0,G_1,其中,k 个任务 $\tau_1,\tau_2,\ldots,\tau_k$ 属于系统任务组 G_0,n–k 个任务 $\tau_{k+1},\tau_{k+2},\ldots,\tau_n$ 属于应用任务组 G_1.对于系统任务组 G_0,采用基于固定优先级的抢占式调度,优先级与抢占阈值相等.对于应用任务组 G_1,依据能量约束判断条件选择不同的调度策略:在非能量约束情况下,采用 PTSI 算法;在能量约束情况下,采用 BSRTS 算法.下面给出两个分组的 GATS 算法.

算法 3. 基于分组的自适应调度算法(GATS).

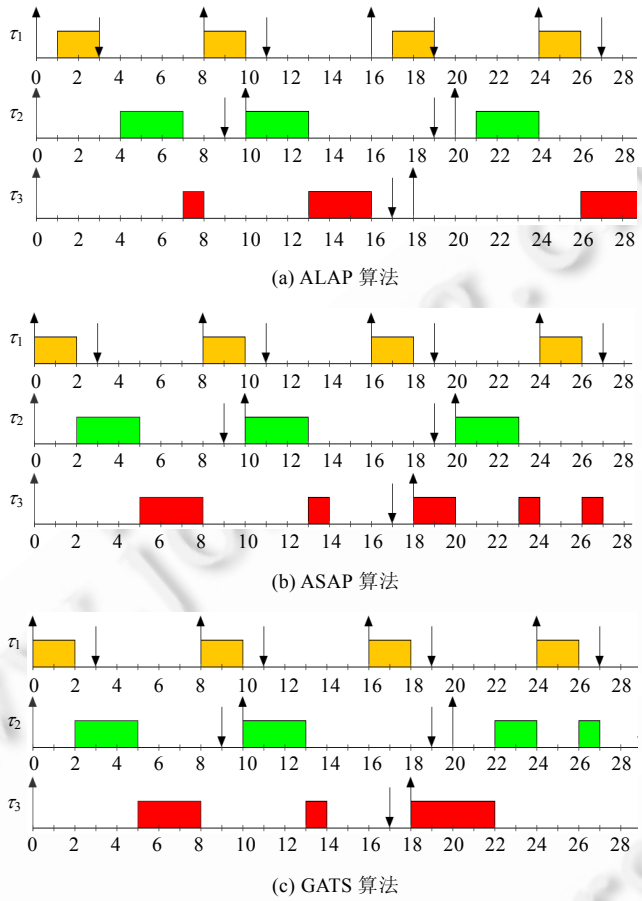
```
1.  //先依据分组规则进行分组,进入调度器后选择策略,next 表示从就绪队列选取的切换任务.
2.  t←0
3.  loop
4.  W←{τ1,τ2,…,τn}                          //就绪队列按固定优先级排序
5.  If W≠∅ then
6.  next←τ1                                  //取任务集合 W 中最高优先级任务
7.     //根据优先级和标示,对任务分组
8.     If next∈[0,low) && Tflag=0
9.        G0←next                            //将任务 τi 加入分组 G0
10.    Else
11.       G1←next                            //将任务 τi 加入分组 G1
12.    Endif
13.    //进入调度器入口
14.    If next∈G0
15.       固定优先级抢占式调度
16.    ElseIf 满足公式(5)                    //任务属于分组 G1,判断能量约束条件
17.       PTSI 算法(算法 1)                  //非能量约束情况
18.    Else
19.       BSRTS 算法(算法 2)                 //能量约束情况
20.    Endif
21. Endif
22. t←t+1
23. Endloop
```

GATS 算法为了避免应用任务的调度策略影响系统任务的正常运行,将系统任务单独分为一组,不改变它们的调度策略;而且系统任务拥有高于应用任务的优先级,保证其实时性.我们假设系统任务分组 G_0 运行的最小需求能量为 $E_{\min}$,用来维持系统的基本运行,但无法执行应用任务.当应用任务分组 G_1 运行的能量消耗需求小

于等于 E_{min} 时,就需要补充能量才能继续执行应用任务.本文主要针对应用任务分组进行任务调度策略分析和实验.

2.3.2 GATS 与 ALAP,ASAP 算法调度对比分析

在非能量约束情况下,图 2 给出了 3 种调度算法的调度对比,调度任务集合属性设置见表 1.



(a) ALAP 算法

(b) ASAP 算法

(c) GATS 算法

Fig.2 Scheduling comparison in non-energy constraint

图 2 非能量约束情况下的调度对比

Table 1 Attributes of tasks set in non-energy constraint

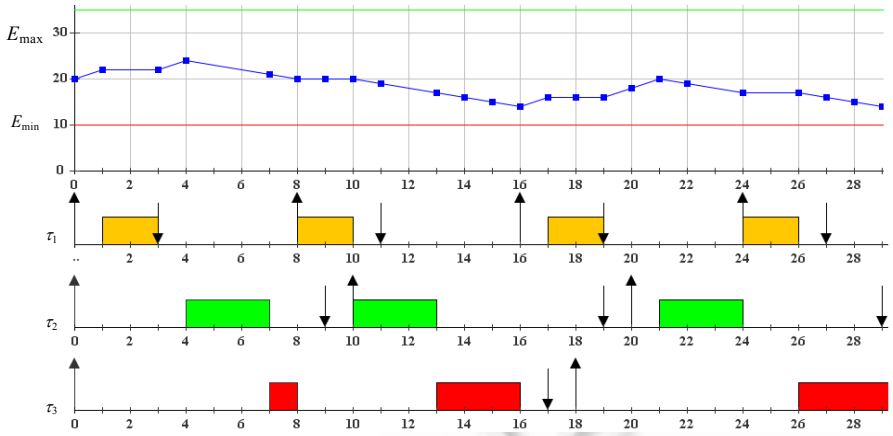
表 1 非能量约束情况下任务集合属性

任务	C_i	E_i	D_i	T_i	π_i	γ_i
τ_1	2	4	3	8	3	3
τ_2	3	9	9	10	6	6
τ_3	4	12	17	18	9	6

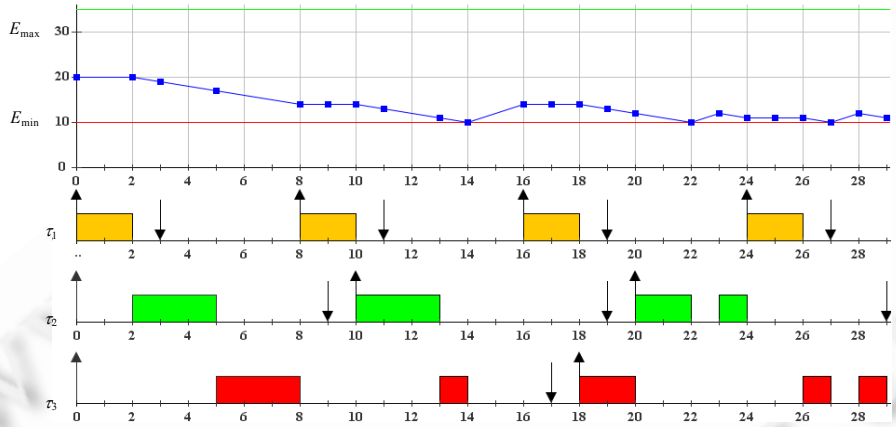
对于 ALAP 算法,如图 2(a)所示,ALAP 算法让任务尽量延迟执行,贡献了大量的空闲时间,但容易造成任务错过截止时间.ASAP 算法如图 2(b)所示,任务一旦释放,就立刻执行,在非能量约束情况下,ASAP 算法等同于固定优先级抢占式调度.如图 2(c)所示,GATS 算法在任务释放时立刻执行,当任务执行时,如果有其他高优先级任务要抢占当前任务,则必须满足其优先级高于当前任务的抢占阈值.例如,在 20s 时,任务 τ_3 正在执行,任务 τ_2 释放,要抢占任务 τ_3,但 τ_2 的优先级等于 τ_3 的抢占阈值,不能抢占 τ_3,这样就避免了 τ_3 被抢占.在一个超周期 360s 内,ASAP 算法发生抢占 25 次,ALAP 算法发生抢占 23 次,GATS 算法发生抢占 21 次.GATS 算法由于抢占阈值

的引入,降低了抢占次数.

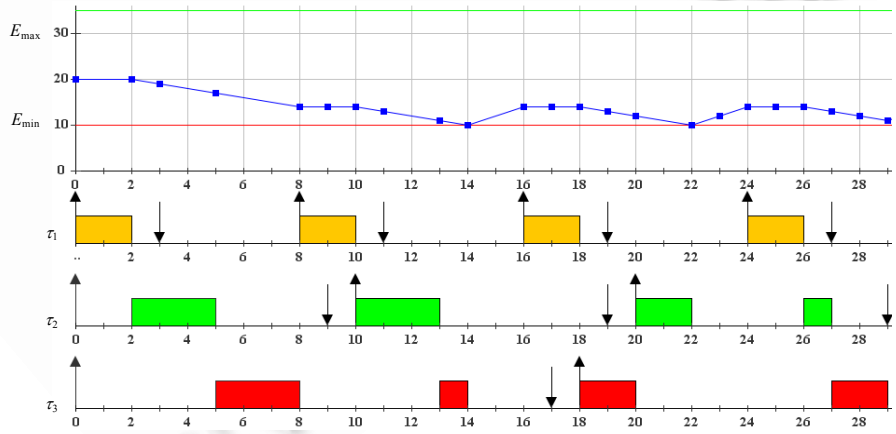
在能量约束情况下,图 3 给出了 3 种调度算法的调度对比,调度任务集合属性设置见表 2.



(a) ALAP 算法



(b) ASAP 算法



(c) GATS 算法

Fig.3 Scheduling comparison in energy constraint

图 3 能量约束情况下的调度对比

Table 2 Attributes of tasks set in energy constraint

表 2 能量约束情况下任务集合属性

任务	C_i	E_i	D_i	T_i	π_i	γ_i
τ_1	2	4	3	8	3	3
τ_2	3	9	9	10	6	6
τ_3	4	12	17	18	9	9

能量属性配置为 $E(0)=20$,$E_{\max}=35$,$E_{\min}=10$,$e_{bat}=2$,运行时长 $Duration=100$.

- 如图 3(a)所示:ALAP 算法尽量推迟任务执行,贡献大量的空闲时间来为电池充电,能够保持电池处于较高的能量水平,但却容易造成任务错过截止时间;
- 对于 ASAP 算法,任务只要释放,并且能量满足当前任务运行,则立刻执行;如果能量不足,则任务空闲 1s,如果能量满足,则立刻执行.如图 3(b)所示:在 22s 时,电池当前能量 $E(t)=E_{\min}=10$,任务 τ_2 空闲 1s,补充 1 个时间单位的能量;23s 时,能量满足,则立刻开始执行;
- GATS 算法尽量保持电池的充电或放电模式,如图 3(c)所示:从第 0s 至第 23s,GATS 算法与 ASAP 算法的充放电模式一致;当 23s 时,有充足的能量运行任务 τ_2,ASAP 算法立刻执行了任务 τ_2,而 GATS 算法计算有 1s 松弛时间,则让电池保持充电状态.这样,GATS 算法就降低了电池频繁的模式切换.GATS 算法还可以降低由于能量不足而引起的上下文切换,例如,在 ASAP 算法中:27s 时,任务 τ_3 由于能量不足,中断运行,等待充电 1s 后开始继续运行;而在 GATS 算法中,由于任务 τ_2 在 22s 后保持了连续 2s 的充电状态,任务 τ_3 在 27s 时并没有中断运行.

在运行时长 100s 内:

- ASAP 发生抢占 20 次,电池模式切换 57 次;
- ALAP 发生抢占 4 次,电池模式切换 38 次;
- GATS 发生抢占 13 次,电池模式切换 21 次.

3 实 验

3.1 实验环境设置

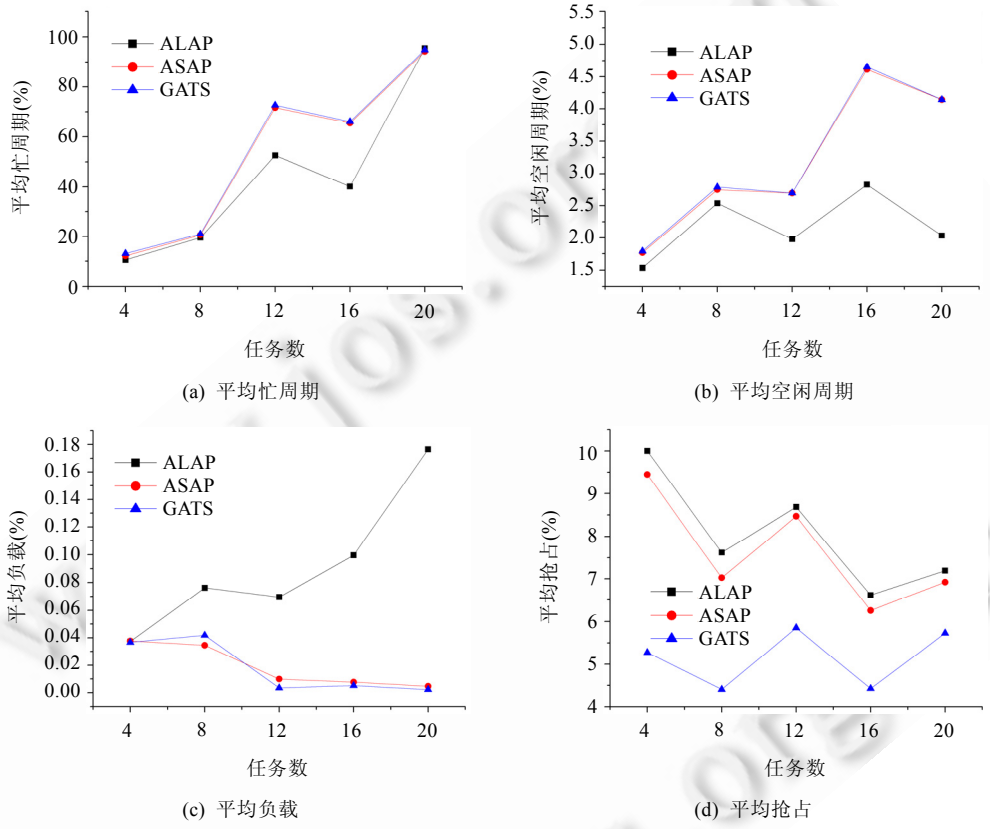
我们利用仿真工具 Yartiss[26]实现了基于分组的自适应调度算法,该工具提供的仿真架构可以依据不同的能量参数和不同算法运行大量任务集合.我们将能量收集单元的能量输出等同于能量补充速率 e_{bat},每个时间单位为电池补充常数个单位能量.任务的能量消耗是线性的,一个任务每执行一个时间单位消耗 E_i/C_i 个单位能量.为了评估算法的性能,我们将 GATS 算法与 ASAP 算法和 ALAP 算法进行对比:在非能量约束情况下,我们通过逐步增加任务数来评估算法性能;在非能量约束情况下,我们通过设计 6 个电池容量场景来评估算法性能,设置 $P_h=e_{bat}=15$,$E_{\min}=0$,运行时长 $Duration=2550$.在仿真过程中,我们使用以下 6 个评价指标,这些指标反映了算法的性能.在非能量约束情况下,我们选择平均忙周期、平均空闲周期、平均负载和平均抢占等指标;在能量约束情况下,我们选择平均忙周期、平价空闲周期、平均负载、平均抢占、平均能量水平和平均电池模式切换等指标.

1) 平均忙周期:指的是仿真期间处理器活动的平均周期.忙周期越长,CPU 利用率越高;
2) 平均空闲周期:指的是仿真期间处理器处于空闲的平均周期.在能量约束情况下,它包括空闲时间和松弛时间.空闲周期越长,电池平均能量水平越高,则系统具有更低的能量约束;
3) 平均负载:指的是仿真期间,平均执行一个调度事件所花费的时间.平均开销越大,越可能造成任务错过任务截止时间;
4) 平均抢占:指的是抢占事件占事件总数的比率.抢占次数越大,上下文切换次数越多,负载增大,性能降低,在实际应用中会造成算法不可用;
5) 平均能量水平:指的是电池或电容在仿真期间的平均能量百分比.平均能量级越高,则系统具有更低的能量限制;

6) 平均电池模式切换:指的是电池模式切换次数占事件总数的比率.电池模式切换次数越多,电池能量利用率越低,在实际应用中会造成系统不能正常工作.

3.2 结果分析

在非能量约束情况下,如图 4 所示,ALAP 算法尽可能地推迟任务执行,贡献了大量的空闲时间,其忙周期和空闲周期最小.但是随着任务数量的增加,ALAP 算法由于计算推迟时间而造成平均负载明显增大.ASAP 算法在非能量约束情况下与固定优先级抢占调度一致,GATS 算法基于固定优先级抢占调度增加了抢占阈值,因此,ASAP 算法与 GATS 算法对平均忙周期、平均空闲周期和平均负载的影响差别不大.GATS 算法由于抢占阈值的引入,在保证任务截止时间的前提下,避免了高优先级任务的抢占,故其抢占次数最少.



(a) 平均忙周期 (b) 平均空闲周期

(c) 平均负载 (d) 平均抢占

Fig.4 Experiment comparison in non-energy constraint

图 4 非能量约束情况下实验指标对比

在能量约束情况下,如图 5 所示,GATS 算法具有更少的抢占次数.这是由于 ASAP 算法每执行一个时间单位就判断抢占,并判断是否有足够的能量:当能量不足时,补充一个时间单位能量,再判断能量是否满足任务执行;当能量满足时,任务立刻执行.而 GATS 算法依据能量水平让任务集中运行,当能量不足时,集中可用的松弛时间来补充能量,减少了由于能量不足而引起的抢占.GATS 算法通过减少电池模式切换,充分利用了松弛时间来补充能量,最大化了忙周期和空闲周期,因此,GATS 算法贡献了较高的能量水平,可降低系统能量约束.ALAP 算法、ASAP 算法和 GATS 算法的平均开销随着电池容量的增大趋于一致.我们发现:随着能量存储单元能量逐渐增大,系统的能量约束变小,任务抢占次数趋于平稳.GATS 算法尽量保持电池充放电模式,较 ASAP 算法减少了电池模式的切换.

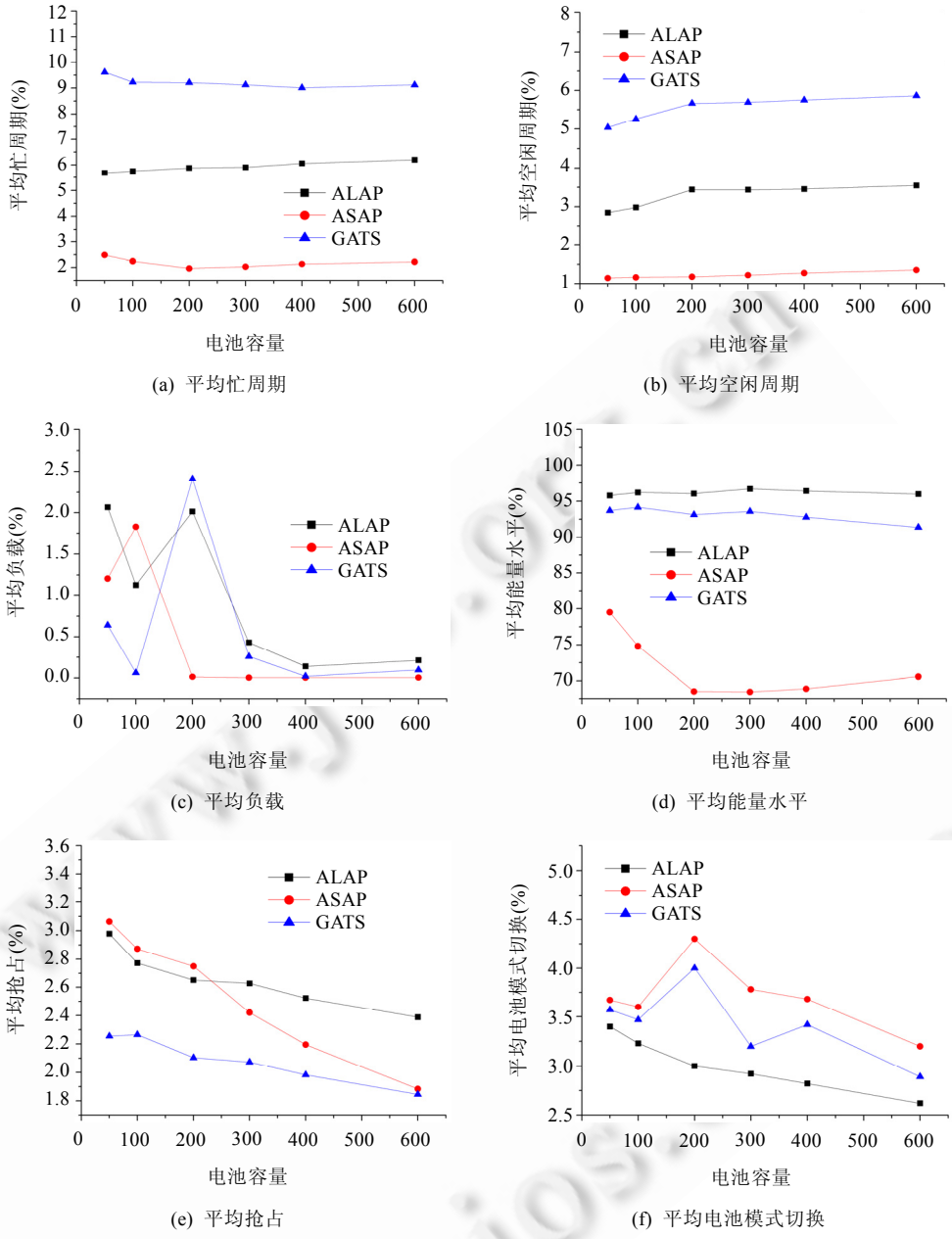


Fig.5 Experiment comparison in energy constraint

图 5 能量约束情况下实验指标对比

4 结束语

本文准确地观察到EHES存在非能量约束和能量约束两种情况,并认为两种不同情况下系统性能和能量对调度算法具有不同的需求.本文通过建立能量约束判断条件,将问题分解为两部分:非能量约束和能量约束,降低了任务调度算法设计的复杂度.在非能量约束情况下,通过减少上下文切换次数来优化系统性能;在能量约束情况下,通过减少电池充放电模式的切换来降低系统能量约束力.我们通过大量的仿真实验来评测 GATS 算法

的有效性,并与 ALAP 算法和 ASAP 算法进行比较.实验结果表明:在能量约束情况和非能量约束情况下,ASAP 算法和 ALAP 算法都具有抢占次数过多的缺点,而 GATS 算法可以很好地克服这个缺点.在能量约束情况下,通过减少电池模式的切换,使得电池保持了较高的平均能量水平,降低了系统的能量约束.

致谢 在本文的写作过程中,杨春鑫同学对任务调度算法理论给予了宝贵的建议;孙朋朋、王征等同学对算法实现和实验做了大量的工作.在此,我们向以上同学表示感谢.

References:

[1] Chen X, Xu Z, Kim H, Gratz PV, Hu J, Kishinevsky M, Oqras U, Ayoub R. Dynamic voltage and frequency scaling for shared resources in multicore processor designs. In: Proc. of the 50th Annual Design Automation Conf. IEEE Press, 2013. 114. [doi: 10.1145/2463209.2488874]

[2] Lin X, Wang Y, Yue S, Chang N, Pedram M. A framework of concurrent task scheduling and dynamic voltage and frequency scaling in real-time embedded systems with energy harvesting. In: Proc. of the 2013 IEEE Int'l Symp. on Low Power Electronics and Design (ISLPED). IEEE Press, 2013. 70−75. [doi: 10.1109/ISLPED.2013.6629269]

[3] Saha S, Ravindran B. An experimental evaluation of real-time DVFS scheduling algorithms. In: Proc. of the 5th Annual Int'l Systems and Storage Conf. New York: ACM Press, 2012. 7. [doi: 10.1145/2367589.2367604]

[4] Dargie W. Dynamic power management in wireless sensor networks: State-of-the-Art. Sensors Journal, 2012,12(5):1518−1528. [doi: 10.1109/JSEN.2011.2174149]

[5] Huang K, Santinelli L, Chen JJ, Thiele L, Buttazzo GC. Applying real-time interface and calculus for dynamic power management in hard real-time systems. Real-Time Systems, 2011,47(2):163−193. [doi: 10.1007/s11241-011-9115-z]

[6] Qiu Q, Liu S, Wu Q. Task merging for dynamic power management of cyclic applications in real-time multi-processor systems. In: Proc. of the Int'l Conf. on Computer Design. IEEE Computer Society Press, 2006. 397−404. [doi: 10.1109/ICCD.2006.4380847]

[7] Kansal A, Hsu J, Zahedi S, Srivastava MB. Power management in energy harvesting sensor networks. ACM Trans. on Embedded Computing Systems (TECS), 2007,6(4):32. [doi: 10.1145/1274858.1274870]

[8] Raghunathan V, Kansal A, Hsu J, Friedman J, Sivastava M. Design considerations for solar energy harvesting wireless embedded systems. In: Proc of the 4th Int'l Symp. on Information Processing in Sensor Networks. Los Alamitos: IEEE Computer Society Press, 2005. 64. [doi: 10.1109/IPSN.2005.1440973]

[9] Allavena A, Mosse D. Scheduling of frame-based embedded systems with rechargeable batteries. In: Proc. of the Workshop on Power Management for Real-time and Embedded Systems (in conjunction with RTAS 2001). 2001. http://igm.univ-mlv.fr/~masson/pdfANDps/allavena_mosse_01.pdf

[10] Moser C, Brunelli D, Thiele L, Benini L. Lazy scheduling for energy harvesting sensor nodes. In: Proc. of the 15th Working Conf. on Distributed and Parallel Embedded Systems, DIPES. New York: Springer-Verlag, 2006. 125−134. [doi: 10.1007/978-0-387-39362-9_14]

[11] Jayaseelan R, Mitra T, Li X. Estimating the worst-case energy consumption of embedded software. In: Proc. of the 12th IEEE Real-Time and Embedded Technology and Applications Symp. Washington: IEEE Computer Society Press, 2006. 81−90. [doi: 10.1109/RTAS.2006.17]

[12] Chandarli Y, Abdeddaïm Y, Masson D. The fixed priority scheduling problem for energy harvesting real-time systems. In: Proc. of the 18th Int'l Conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA). Washington: IEEE Computer Society Press, 2012. 415−418. [doi: 10.1109/RTCSA.2012.72]

[13] Abdeddaïm Y, Chandarli Y, Masson D. Toward an optimal fixed-priority algorithm for energy-harvesting real-time systems. In: Proc. of the Work in Progress Session of the 19th IEEE Real-Time and Embedded Technology and Applications Symp. 2013. 45−48. https://hal.archives-ouvertes.fr/hal-00796646/document

[14] Abdeddaïm Y, Chandarli Y, Masson D. The optimality of PFPasap algorithm for fixed-priority energy-harvesting real-time systems. In: Proc. of the 25th Euromicroc Conf. on Real-Time Systems (ECRTS). IEEE Press, 2013. 47−56. [doi: 10.1109/ECRTS.2013.16]

[15] Abdeddaïm Y, Chandarli Y, Davis RI, Masson D. Approximate response time for fixed priority real-time systems with energy-harvesting. Technical Report, 2014. https://hal.archives-ouvertes.fr/hal-00986340/

[16] Liu S, Qiu Q, Wu Q. Energy aware dynamic voltage and frequency selection for real-time systems with energy harvesting. In: Proc. of the Design, Automation and Test in Europe. IEEE Press, 2008. 263−241. [doi: 10.1109/DATE.2008.4484692]

[17] Liu S, Lu J, Wu Q, Qiu Q. Harvesting-Aware power management for real-time systems with renewable energy. IEEE Trans. on Very Large Scale Integration (VLSI) Systems, 2012,20(8):1473−1486. [doi: 10.1109/TVLSI.2011.2159820]

[18] Patel MR, Wrote; Han B, Chen Q, Cui XT, Trans. Spacecraft Power Systems. Beijing: China Astronautic Publishing House, 2010. 60−70 (in Chinese).

[19] Wang ZJ, Xie LL. Cyber-Physical systems: A survey. Acta Automatic Sinica, 2011,37(10):1157−1166 (in Chinese with English abstract).

[20] Banerjee A, Venkatasubramanian KK, Mukherjee T, Gupta SK. Ensuring safety, security, and sustainability of mission-critical cyber-physical systems. Cyber-Physical Systems, 2012,100(1):283−299. [doi: 10.1109/JPROC.2011.2165689]

[21] Liu S, Lu J, Wu Q, Qiu Q. Load-Matching adaptive task scheduling for energy efficiency in energy harvesting real-time embedded systems. In: Proc. of the 16th ACM/IEEE Int'l Symp. on Low Power Electronics and Design. IEEE Press, 2010. 325−330. [doi: 10.1145/1840845.1840912]

[22] Tsafrir D. The context-switch overhead inflicted by hardware interrupts (and the enigma of do-nothing loops). In: Proc. of the 2007 Workshop on Experimental Computer Science. New York: ACM Press, 2007. [doi: 10.1145/1281700.1281704]

[23] Wang Y, Saksena M. Scheduling fixed-priority tasks with preemption threshold. In: Proc. of the 6th Int'l Conf. on Real-Time Computing Systems and Applications. Los Alamitos: IEEE Press, 1999. 328−335. [doi: 10.1109/RTCSA.1999.811269]

[24] Gao F, Yang K, Hui D, Li DH. Cycle-Life energy analysis of $LiFePO_4$ batteries for energy storage. Proc. of the CSEE, 2013,33(5): 41−45 (in Chinese with English abstract).

[25] Chetto M, Masson D, Midonnet S. Fixed priority scheduling strategies for ambient energy-harvesting embedded systems. In: Proc. of the 2011 IEEE/ACM Int'l Conf. on Green Computing and Communications (GreenCom). IEEE Computer Society , 2011. 50−55. [doi: 10.1109/GreenCom.2011.17]

[26] Chandarli Y, Fauberteau F, Masson D, Midonnet S, Qamhieh M. Yartiss: A tool to visualize, test, compare and evaluate real-time scheduling algorithms. In: Proc. of the 3rd Int'l Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems. 2012. 21−26. https://hal-upec-upem.archives-ouvertes.fr/hal-00691985/document

附中文参考文献:

[18] Patel MR,著;韩波,陈琦,崔晓婷,译.航天器电源系统.北京:中国宇航出版社,2010.60−70.

[19] 王中杰,谢璐璐.信息物理融合系统研究综述.自动化学报,2011,37(10):1157−1166.

[24] 高飞,杨凯,惠东,李大贺.储能用磷酸铁锂电池循环寿命的能量分析.中国电机工程学报,2013,33(5):41−45.

葛永琪(1980－),男,宁夏青铜峡人,博士生,主要研究领域为嵌入式系统能量管理,实时任务调度.

张健(1990－),男,硕士,主要研究领域为嵌入式系统能量管理,实时任务调度.

董云卫(1968－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为嵌入式系统,信息物理融合系统,可信软件设计与验证.

顾斌(1968－),男,研究员,CCF 高级会员,主要研究领域为可信软件,计算机控制.