能耗限制的松弛任务实时调度算法

桂盛霖，雷 航

(电子科技大学计算机科学与工程学院 成都 610054)

【摘要】许多嵌入式系统依靠单电池供电。基于嵌入式实时操作系统，提出了一种具有通用性的松弛任务模型，并在此基础上根据电池剩余电能设计了相应的实时调度算法来防止在任务运行过程中耗尽不可充电电源的电能所引起的失效。结果表明，该模型能够有效及时地根据剩余电池电能来调整任务周期，使之能够在使命时间内正确完成计算任务，在一定程度上解决了电池电能对使命时间的约束问题。

关 键 词 动态电压调整； 能耗限制； 松弛任务； 实时调度

中图分类号 TP301.6 **文献标识码** A

On Energy-Constrained Real-Time Scheduling for Elastic Tasks

GUI Sheng-lin and LEI Hang

(School of Computer Science and Engineering, University of Electronic Science and Technology of China Chengdu 610054)

Abstract The energy of many modern embedded systems is power-supplied by the batteries. Based on real-time operating system, this paper presents an elastic task model on energy-constrained with universalization, then designs a corresponding real-time scheduling algorithm for guaranteeing the system stabilization and avoiding failtures caused by running out the energy of the non-recharge power-supply according to the rest energy of the battery. The result shows that this model could change the periods of taks according to the rest energy of the battery effectively and timely, so the computation could be completed in their mission times.

Key words dynamic voltage scaling; energy-constrained; elastic tasks; real-time scheduling

随着微处理器向性能更高、更智能的方向发展，在追求更高性能、更快处理速度的同时，是以消耗更多的能耗为代价。在很多场合下使用的嵌入式系统是由充电电池供电，其持续使用时间是有限的。目前，在国外有很多学者和机构都在进行受能耗限制的实时调度研究。文献[1]提出了一种动态窗口限制调度(dynamic window constrained scheduling)模型，主要考虑能耗和系统动态失效的函数关系。文献[2]证明了在一个有限能耗预算并且具有离散速度的DVS能力的系统上，能否满足所有固定周期任务的截止限是个NP-Hard问题，但没有提出明确的任务模型。文献[3]主要介绍了动态电压调整(dynamic voltage scaling)，但没有考虑硬能耗限制。

本文重点讨论了具有通用性的松弛任务模型[4]下的能耗问题，通过建立能耗模型，详细介绍了受能耗限制的具有松弛任务周期的实时调度算法。

在某些嵌入式实时应用中，在单电池供电的系统中的任务可以在不同的条件下以不同的频率被执行。如在检测生态环境变化的传感器网络(wireless sensor network)中，每个传感器节点由单电池供电，如何延长其工作时间已成为研究热点[5]。传感器节点的采样频率越高，能耗越大，其持续工作时间也就越短。如在保证最低采样频率(每20 min采样一次)的前提下，每10 min采样一次与每15 min采样一次，除了提高部分数据精度外，没有本质的区别，但前者的能耗大于后者。工作时间小于后者。因此，引入松弛任务模型下的能耗管理以满足在对能耗和工作时间要求均较高的实时应用及非精确计算中[6]。

1 能耗限制的松弛任务模型

1.1 系统模型

模型假设：

(1) 系统是一个依靠电池电源供电的单处理器系统。为了保持系统的稳定性，电源能源提供应该保持在一个安全的范围$[E_{\min}, E_{\max}]$，其中$E_{\max}$表示电源的最大能源容量，$E_{\min}$表示可保证稳定操作的最低门槛限度。因此，系统的有限能耗预算是$E_{\text{budget}}=E_{\max}-E_{\min}$。

收稿时间：2006－12－16； 修回日期：2007－08－24
基金项目：国家863计划重点项目(2007AA010304)；国家自然科学基金项目(90718019)
作者简介：桂盛霖(1983－)，男，博士生，主要从事嵌入式操作系统技术方面的研究.

(2) 在[0, X]时间区间，系统必须保持可操作性，即CPU能源消耗在X时间单位内不能超过E_{budget}。将时间间隔[0, X]称为使命时间(mission time)。在使命时间间隔内对电池进行充电是不可能的，只有在使命时间结束后才能对电池进行充电。

(3) 系统不具有DVS(dynamic voltage scaling)能力(文献[7-9]介绍了具有DVS能力的系统的能耗调度)，系统只能工作在执行模式或者空闲模式。

在空闲模式下，CPU不执行任何任务，消耗较少的能源。如果g^{stb}表示消耗能源；$g(T_{i,j})$表示任务τ_i的周期为T_j时执行所消耗的能耗,则在任何时间区间[t_1, t_2]，总的能耗为：

$$E(t_1,t_2)=\sum_{i=1}^{n}E_i=\sum_{i=1}^{n}\sum_{j=1}^{m}g(T_{i,j})\,t_{i,j}+g^{\text{stb}}t_{\text{stb}} \tag{1}$$

式中　$t_{i,j}$和t_{stb}分别表示任务τ_i的周期为T_j时的执行时间和空闲模式下所花费的总时间，$t_{i,j}\leqslant t_2-t_1$。

(4) 能耗调度算法运行的能耗开销忽略不计。

1.2　任务模型

定义松弛周期任务集为$\Gamma=\{\tau_1, \tau_2,\cdots, \tau_n\}$。每个任务$\tau_i$的周期$T_i$等于它的截止限。允许任务间剥夺，采用EDF调度策略[10]。每个松弛任务表示为$\tau_i(C_i, T_{i_0}, T_{i_{\min}}, T_{i_{\max}}, e_i)$，其中$C_i$表示任务的最坏执行时间；$T_{i_0}$表示名义周期(nominal periods)；$T_{i_{\min}}$表示任务$i$的最小周期；$T_{i_{\max}}$表示任务$i$的最大周期；$e_i$表示松弛系数。由于采用EDF调度，则必须保证$C_1/T_{1_{\min}}+C_2/T_{2_{\min}},\cdots,+C_n/T_{n_{\min}}\leqslant 1$。松弛系数$e_i$表示任务$\tau_i$改变它自己的周期的灵活度。系数$e_i$的存在说明了任务不可随意改变自己的周期：$e_i$越大，任务越松弛；如果$e_i$为0，则表示任务的周期不允许缩放。事实上，如果一个任务的$T_{i_0}=T_{i_{\min}}=T_{i_{\max}}$，并且$e_i$=0，那么它就退化成为一个具有硬时限的固定周期任务。因此，用本模型建模具有很强的通用性。当系统初始化时，令$T_{i_0}=T_{i_{\min}}$，如果$T_{i_{\min}}\leqslant T_{i_0}<T_{i_{\max}}$，则表示任务$\tau_i$处于降级服务阶段。$T_{i_{\max}}$是任务执行的最小频率，也是任务$\tau_i$降级服务的低限。一旦任务$\tau_i$的周期等于$T_{i_{\max}}$系统仍然无法调度，则可能会带来灾难性的后果。本文所提出的调度算法，正是根据以上模型假设，动态缩放任务集周期，以使任务集在能耗限制内达到最大的运行效率。

2　能耗限制的实时调度算法

由于系统必须在时间间隔[0，X]内保持可操作性，应确定能否在有限能耗的条件下满足所有任务的截止时间。因此需要确定在上述条件下完成操作需要的最小CPU能耗。由于任务的周期是可变的，要完成操作需要的最小CPU能耗必须要大于或等于当所有任务处于最小周期时执行所需要的能耗，将该门槛能耗记为$E_{\text{threshold}}$。

定义　如果一个实时系统的$E_{\text{budget}}<E_{\text{threshold}}$，那么该实时系统是受能耗限制的。

如果系统受能耗限制，则以所有任务的最小周期来执行所有的任务，就可能在使命时间结束前耗尽能源，从而给使用者造成危害。因此，要在使命时间结束前保持系统的可操作性，最大化系统性能。由于每个任务的周期可变，则$\lfloor X/T_{i_{\min}}\rfloor\geqslant\lfloor X/T_{i_{\max}}\rfloor$，那么每个任务$\tau_i$在$T_{i_{\min}}$条件下所花费的能耗大于等于在$T_{i_{\max}}$条件下所花费的能耗。所以$T_i$与能耗成反比例。为简单起见，定义当任务$\tau_i$的周期为$T_{i_0}$时，$g(T_{i_0})=F(T_{i_0}))=1/T_{i_0}$。假定每个任务$\tau_i$的周期变为$T_i$时，总能耗$g(b)$能够满足$E_{\text{budget}}$。

2.1　调度算法

所有任务均以名义周期执行的总能耗表示为：

$$g(T_0)=\sum_{i=1}^{n}g(T_{i_0})=\sum_{i=1}^{n}1/T_{i_0} \tag{2}$$

令W为每个任务τ_i的周期压缩率，表示为：

$$\forall i\ W=(1/e_i)\left|1/T_{i_0}-1/T_i\right| \tag{3}$$

式中　绝对值符号当$T_{i_0}<T_i$取正；反之，绝对值取负。因此由式(2)和式(3)可以求得未知的$T_1,T_2,\cdots,T_n$：

$$\forall i\ T_i=1/\left((1/T_{i_0})-(g(T_0)-g(b))\times e_i/\sum_{i=1}^{n}e_i\right)=$$

$$1/\left((1/T_{i_0})-(g(T_0)-E_{\text{budget}})\times e_i/\sum_{i=1}^{n}e_i\right) \tag{4}$$

每一个任务的周期变化都必须控制在[$E_{\min}$, $E_{\max}$]内，也就是说当某些任务的周期变为$T_{i_{\max}}$时，总能耗仍然大于E_{budget}时，那么此时只能选择那些$T_{i_0}\leqslant T_{i_{\max}}$的任务执行式(4)。因此，任务集$\Gamma$可以分成2个子集：子集$\Gamma_a$包括所有当前任务周期已经是$T_{i_{\max}}$的任务，其总能耗表示为$g_a(T_{\max})$；子集$\Gamma_s$包括所有当前任务周期小于$T_{i_{\max}}$的任务，其总能耗表示为$g_s(T_0)$。由式(4)可得：

$$\forall\tau_i\in\Gamma_s$$

$$T_i=1/\left((1/T_{i_0})-(g_s(T_0)-E_{\text{budget}}+g_a(T_{\max}))\times e_i/K_v\right) \tag{5}$$

式中　$K_v=\sum_{\tau_i\in\Gamma_s}e_i$。如果通过式(5)计算出$T_i>T_{i_{\max}}$，则将$T_i$赋值为$T_{i_{\max}}$，并更新子集合$\Gamma_s$和$\Gamma_a$，即在子集

合Γ_s中删除τ_i元素，在子集合Γ_a中添加τ_i元素。经过若干次上述计算之后，若所有任务都能调整到适当的周期使总能耗小于等于E_{budget}，则称该任务集Γ是可调度的。

基于上述模型和公式推导，具体算法如下：

(1) 为每个任务τ_i的T_{i_0}赋初值$T_{i_{\min}}$，根据式(2)计算出$g(T_0)$和$g(T_{\max})$，以及给定E_{budget}的值。

(2) 如果E_{budget}小于$g(T_{\max})$或$C_1/T_{1_{\min}}+C_2/T_{2_{\min}}+\cdots+C_n/T_{n_{\min}}>1$则该任务集不可调度，返回FAIL结果。如果$E_{\text{budget}}$大于$g(T_{\min})$，则每个任务都可以最小周期执行，此时不存在能耗限制，跳转到步骤(5)执行。

(3) 判断每个任务的e_i和T_{i_0}，如果任务τ_i的e_i=0或者$T_{i_0}=T_{i_{\max}}$，则把该任务加入到集合Γ_a中，否则加入到集合Γ_s中。

(4) 根据式(5)计算出T_i，如果$T_i\geqslant T_{i_{\max}}$，则将$T_{i_0}$赋值为$T_{i_{\max}}$，并将$\tau_i$加入到集合$\Gamma_a$。在集合$\Gamma_s$中删除掉$\tau_i$，然后又转入步骤(4)继续计算其他任务的$T_j$；如果$T_i<T_{i_{\max}}$，则将$T_{i_0}$赋值为$T_i$，然后转入步骤(4)继续计算其他任务的$T_j$。

(5) 返回FEASIBLE结果和所有任务的T_{i_0}。

以上算法是针对在能耗受限的情况下，兼顾能耗和效率而设计的。在能耗受限的情况下，能够准确快速地计算出满足能耗的各任务的周期值且算法本身具有很高的效率，其时间复杂度仅为$O(n)$。根据计算出的任务集周期值调整任务集的周期，一定能够保证在使命时间内不会出现电能耗尽的情况，从而维持系统的稳定。

任务集中每个任务受系数e_i限制，不可随意调整周期。而实际情况有可能是在使命时间$[0, X]$内有某周期任务τ_j因为计算完成而被终止，则其他剩余任务可以在剩下的能耗允许范围内根据e_i缩短周期，获得更大的效率。这种情况也可以利用上述算法进行处理。只需在终止该任务的时刻，计算出已用时间占使命时间的比例c、$g(T_{j_{\min}})$和剩下任务的$g(T_0')$和$g(T_{\max}')$，然后将$g(T_0')$、$g(T_{\max}')$和$E_{\text{budget}}'=E_{\text{budget}}-c\times g(T_{j_{\min}})$分别代入上述算法的$g(T_0)$、$g(T_{\max})$和$E_{\text{budget}}$即可。最简单的情况就是$g(T_{\max}')\leqslant E_{\text{budget}}'$，那么所有剩下的任务都可以最小周期运行。

2.2 仿真实验及分析

如果将该算法与嵌入式实时操作系统的任务调度器紧耦合使用，则可以形成更完善的调度方案。任务调度器在任务集Γ创建时或任务集Γ中某任务被创建/终止时被激活。一旦被激活，任务调度器就计算每个任务的新周期，并在该任务的下一次释放时间(release time)更改。在此设计了两个仿真实验，分别针对当有任务中途执行完毕时和当有一个任务中途被创建时两种情况，以说明算法的有效性及可靠性。

实验 1 3个松弛周期任务在时刻t=0被同时创建。任务参数集如表1所示。

表1 实验1数据

任务	C_i/ms	T_{i_0}/ms	$T_{i_{\min}}$/ms	$T_{i_{\max}}$/ms	e_i
τ_1	20	100	100	200	2
τ_2	18	60	60	100	1.5
τ_3	30	60	60	80	0.5

由定义可知，要让系统成为能耗受限系统，必须满足$E_{\text{budget}}<E_{\text{threshold}}$。计算可知$E_{\text{budget}}$必须满足不等式$27.5\leqslant E_{\text{budget}}<43.34$。因此假定$E_{\text{budget}}$的值为33。在时刻$t$=2同时释放该3个任务，在时刻$t$=7任务$\tau_2$由于执行完毕被终止。为了在图1中表示时间的方便和清晰，假定时间段[0,7]占使命时间的比例c等于1/3。由于受能耗的限制，调度器在以上两个时刻t=2和t=7作出周期调整，已经执行完毕的任务实例总数和时间之间的函数关系如图1所示。设计本实验的目的是为了说明任务集在开始被调度时，调度算法能够计算任务集所需能耗，如果所需能耗大于E_{budget}，则减小任务集任务的执行频率使所需能耗小于等于E_{budget}，这样确保在整个不可充电的使命时间内绝对不会发生电能耗尽的情况；在当有任务执行完成退出时，算法能够动态计算出剩余电能，适当增加剩余任务的执行频率。文献[1]中所提出的DWCS模型是一种尽力而为模型，并不能有效保证在使命时间内不发生失效。因此，本文所提出的模型的可靠性和安全性明显高于文献[1]中的DWCS模型。

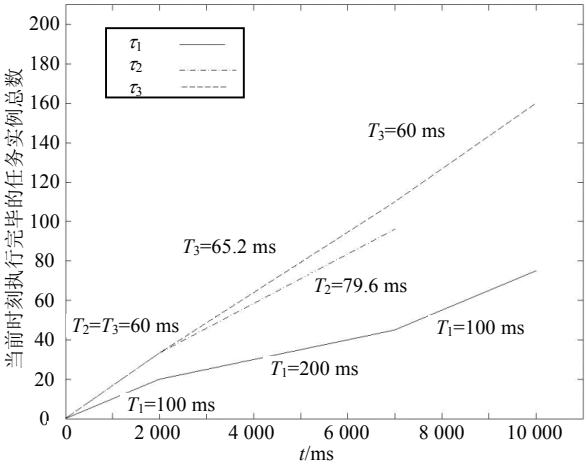

图1 同时调度任务集任务时和当有一个任务退出时的任务集周期的动态调整周期过程

实验 2　与实验1相似，假设给定E_{budget}的值为32.25，松弛周期任务τ_1和τ_2从时刻t=0开始执行。任务参数集如表2所示。

表2　实验2数据

任务	C_i/ms	T_{i0}/ms	$T_{i\min}$/ms	$T_{i\max}$/ms	e_i
τ_1	15	100	100	150	1
τ_2	18	60	60	90	1.5
τ_3	40	80	80	105	0.5

在时刻t=5时，任务τ_3开始释放，假定c_1=c_2=1/10，调度器在时刻t=5作出调整，已经执行完毕的任务实例总数和时间之间的函数关系如图2所示。

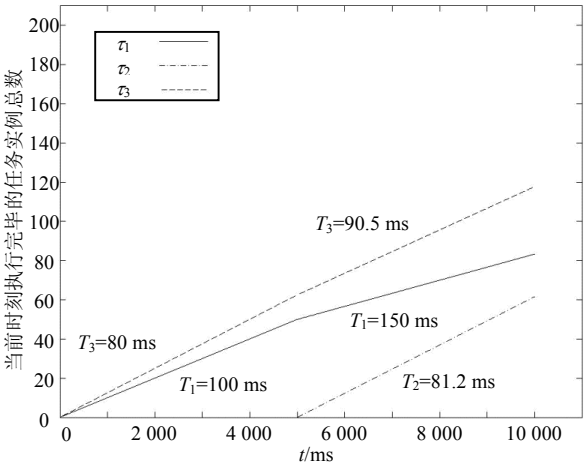

图2　在任务执行过程中有新建任务加入时任务集周期动态调整过程

实验的目的是为了说明算法能够动态保证当前执行的任务集获得最大的执行效率：从时刻t=0开始执行任务τ_1和τ_2，算法调整τ_1和τ_2的周期使其能耗小于等于E_{budget}，在时刻t=5任务τ_3就绪，因此再次调用算法调整三个任务的周期，使能耗小于等于系统剩下的电能，以确保在整个不可充电的使命时间内绝对不会发生电能耗尽的情况。

4　结　论

本文所提出的模型相比文献[1]所提出的窗口模型具有更好的通用性和可靠性。文献[1]所提出的模型并不能保证使命时间内不出现危及到系统稳定性的失效，可靠性很低。前面已经说明，具有固定周期的周期任务是松弛周期任务的一种特例，因此本文所设计的能耗算法也就具有一般的通用意义。不论是运行过程中有任务就绪还是终止，只需改变参数，使用本算法框架均可以得到处理。由仿真结果可以看出，本算法相对DWCS算法有更高的准确性和可靠性：应用计算出的周期值，能够保证在使命时间内绝对不会发生电能耗尽的情况，并且由于每个任务的$T_{i\max}$参数的限制，保证了每个任务的最小执行频率，从而使得系统能够保持很好的稳定性、可操作性和持久性，绝对不会发生在使命时间结束前电源提前耗尽的情况并且保证了系统的效率。故本算法可以和嵌入式操作系统调度器紧耦合，从而进一步提高系统的可靠性。

参　考　文　献

[1] ALENAWY T A, AYDIN H. Energy-constrained scheduling for weakly-hard real-time systems[C]//In: Proceedings of the 26th IEEE International Real-Time Systems Symposium (RTSS' 05). Miami: IEEE Computer Society, 2005.

[2] ALENAWY T A, AYDIN H. On energy-constrained real-time scheduling[C]//In: Proceedings of the 16th EuroMicro Conference on Real-Time Systems(ECRTS' 04). Catania: IEEE Computer Society, 2004.

[3] PILLAI P, SHIN K G. Real-time dynamic voltage scaling for low power embedded operation systems[C]//In: Proceedings of the 18th ACM Symposium on Operating Systems Principles(SOSP' 01). Banff: ACM, 2001.

[4] BUTTAZZO G C, LIPARI G, ABENI L. Elastic task model for adaptive rate control[C]//In: Proceedings of the IEEE Real-Time Systems Symposium. Madrid: IEEE Computer Society, 1998.

[5] HAN C C, KUMAR R, SHEA R, et al. A dynamic operating system for sensor nodes[C]//Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services, Seattle. Washington: ACM, 2005.

[6] LIU J, LIN K J, SHIH W K, et al. Algorithms for scheduling imprecise computations[J]. IEEE Computer, 1991, 24(5): 58-68,.

[7] FLAUTNER K, REINHARDT S, MUDGE T. Automatic performance-setting for dynamic voltage scaling[C]//In: Proceedings of the 7th Conference on Mobile Computing and Networking MOBICOM'01. Rome: ACM, 2001.

[8] GOVIL K, CHAN E, WASSERMANN H. Comparing algorithms for dynamic speed-setting of a low-power CPU[C]//In: Proceedings of the 1st Conference on Mobile Computing and Networking MOBICOM'95. California: ACM, 1995.

[9] GRUIAN F. Hard real-time scheduling for low energy using stochastic data and DVS processors[C]//In: Proceedings of the International Symposium on Low-Power Electronics and Design ISLPED'01. Huntington Beach: ACM, 2001.

[10] KRISHNA C M, and SHIN K G. Real-time systems[M]. [S.l.]: McGraw-Hill, 2001.

编　辑　熊思亮