ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.3, March 2006, pp.602−610
DOI: 10.1360/jos170602

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

优先级有限时的单处理器静态优先级调度*

王保进[1,2+], 李明树[2], 王志刚[1]

[1](解放军信息工程大学 信息工程学院,河南 郑州 450002)

[2](中国科学院 软件研究所 互联网软件技术实验室,北京 100080)

Uniprocessor Static Priority Scheduling with Limited Priority Levels

WANG Bao-Jin[1,2+], LI Ming-Shu[2], WANG Zhi-Gang[1]

[1](Institute of Information and Engineering, PLA Information and Engineering University, Zhengzhou 450002, China)

[2](Laboratory for Internet Software Technologies, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

+ Corresponding author: Phn: +86-10-62624525 ext 6003, E-mail: baojin_wang@itechs.iscas.ac.cn, http://www.iscas.ac.cn

Wang BJ, Li MS, Wang ZG. Uniprocessor static priority scheduling with limited priority levels. *Journal of Software*, 2006,17(3):602–610. http://www.jos.org.cn/1000-9825/17/602.htm

Abstract: In practice, the schedulability of static priority scheduling may be reduced when priority levels of the system are insufficient. If a task set requires more priority levels than the system can support, several tasks must be assigned the same priority level. This causes the priority mapping problem. To solve it, a priority mapping algorithm and necessary and sufficient conditions for analyzing the schedulability of a task after priority mapping are needed. There are three kinds of priority mapping algorithms: decreasing priority assignment algorithm, increasing priority assignment algorithm, and threshold segment mapping algorithm. This paper presents implementation and analyzing conditions of algorithms. Properties of the algorithms are also described and showed. Performance of the algorithms is compared through simulations. Simulation results and conclusions are useful for designing and implementing real-time embedded systems.

Key words: priority mapping; DPA(decreasing priority assignment) algorithm; IPA(Increasing priority assignment) algorithm; preemption threshold; TSM(threshold segment mapping) algorithm

摘 要: 静态优先级调度在实际应用中经常受到系统支持的优先级个数的影响,当任务个数多于系统优先级个数时,需要将几个任务优先级映射成一个系统优先级.这可能引起优先级映射问题,使映射前可调度的系统(任务集合)在映射后变得不可调度.解决这一问题需要减少时间复杂度的映射算法和判定映射后任务可调度性的充分必要条件主要存在3种映射算法:(1) 按照任务优先级递减顺序进行映射的DPA(decreasing priority assignment)算法;(2) 按照优先级递增顺序进行映射的 IPA(Increasing priority assignment)算法;(3) 阈值段间映射法(threshold segment mapping,简称TSM).描述了3种算法的实现和判定条件,论述并证明了算法特性,分析并通过仿真实验比较了算法的性能,最后总结了3种算法各自的适用场合.比较结果和结论对实时嵌入式系统的设计和实现具有一定的参考价值.

* Supported by the National High-Tech Research and Development Plan of China under Grant No.2002AA1Z2302 (国家高技术研究发展计划(863))

Received 2004-12-17; Accepted 2005-05-18

关键词: 优先级映射;DPA 算法;IPA 算法;抢占阈值;TSM 算法
中图法分类号: TP316 文献标识码: A

与动态优先级调度算法相比,静态算法具有易实现、系统开销小和过载时更好的可预测性等优点,在单处理器实时嵌入式系统中得到广泛应用.现有的静态优先级调度算法[1,2]都假定能使用的系统优先级个数无限多,而实际上底层系统支持的优先级个数是有限的.例如 MSI STD BUS 支持两个优先级[3],实时 POSIX 支持 32 个优先级,VxWorks 支持 256 个优先级.当系统提供的优先级个数不足时,只能给几个任务分配相同的系统优先级(或者称作将几个任务分配到同一系统优先级队列中).一般情况下,实时操作系统按照先到先服务(FIFO)的方式处理同一优先级队列内的任务.所以映射后,同一队列内的低优先级(此处是任务优先级)任务会阻塞高优先级任务的执行,从而形成优先级倒置,可能造成映射前可调度的任务集合,映射后变得不可调度.这就是优先级映射问题.此时需要解决:(1) 以何种方式将几个不同的任务优先级映射成一个系统优先级;(2) 怎样判定映射后任务的可调度性.

文献[3]提出常量法用于优先级映射.该算法容易实现,但映射后使任务集合的可调度性下降很多.文献[4]假定映射后同一队列内的任务按照 FIFO 方式执行,队列间任务按照静态优先级抢占调度规则执行,并据此给出了判定映射后任务可调度性的充要条件.文献[4]并未给出具体的映射算法,而且假定任务截止期小于或者等于任务周期.文献[5]对文献[4]的判定条件进行扩展,给出截止期大于周期时,映射后的判定条件.提出了 FPA 优先级分配算法,按照优先级递减的顺序分配任务到系统优先级队列,队列内和队列间任务的执行方式与文献[4]描述的相同.

文献[6]假定映射后队列间任务按照 RM(rate monotonic)规则调度执行,队列内任务按照随机(RAN)选取的方式执行.给出了判定映射后任务可调度性的充要条件,但未给出具体的映射算法.文献[8]给出了分布式环境下的优先级映射算法——LOFP 算法,按照优先级递增的顺序分配任务到系统优先级队列.判定映射后任务可调度性的充要条件是对文献[4]提出的判定条件的扩展.LOFP 和 FPA 算法都存在映射后任务集合可调度性会降低的缺点.文献[7]给出了分布式系统中静态和动态两种映射算法,但没有给出判定映射后任务可调度性的充要条件.本文提出了利用抢占阈值调度模型[9,10]生成的非抢占组的特性进行优先级映射的 TSM(threshold segment mapping)算法,此时的判定条件与文献[9]描述的相同.TSM 算法能使用比 FPA 和 LOFP 算法更少的系统优先级,并且映射没有引起新的阻塞时间,从而不会造成前面所述的优先级映射问题.但该算法需要一个应用层的事件驱动线程框架来配合.

可以看出,具有判定条件的映射算法包括 FPA,LOFP 和 TSM 算法.LOFP 算法用于分布式系统,本文研究单处理器系统的优先级映射问题,所以将其分配方法应用到单处理器系统,称为 IPA(increasing priority assignment)算法.另外,为了讨论方便,将按照优先级递减顺序分配任务的 FPA 算法称为 DPA(decreasing priority assignment)算法.

本文第 1 节描述 3 种算法的实现.第 2 节介绍算法的判定条件.第 3 节描述 DPA 和 TSM 算法的特性,并证明 IPA 算法的特性.第 4 节比较算法性能,论述各种算法的适用场合.最后作出总结.

1 算法描述

定义$\Gamma=\{\tau_1,\tau_2,\ldots,\tau_n\}$为一个包含 n 个相互独立的周期性或者偶发性(sporadic)任务的集合,集合中的任务用$\tau_i=(T_i,C_i,D_i)(i=1,2,\ldots,n)$表示,其中,$T_i$ 表示 τ_i 的周期(对于偶发性任务就是最小到达间隔),C_i 表示 τ_i 的最坏情况执行时间,D_i 表示 τ_i 的相对截止期.我们假定大的数值表示高的优先级,对于集合Γ,最小的优先级是 1,最大的优先级是 n.

1.1 DPA算法和IPA算法

任务优先级用Π表示:

$\Pi=\{\pi_1,\pi_2,\ldots,\pi_n\},\forall \pi_i\in\Pi(1\le i\le n),\forall \pi_j\in\Pi(1\le j\le n)$,如果 $i\ne j$,则 $\pi_i\ne\pi_j$.

对于某一优先级映射算法 f,定义:

$\forall \pi_i\in\Pi,\forall \pi_j\in\Pi$,如果 $\pi_i<\pi_j\Rightarrow f(\pi_i)<f(\pi_j)$或者 $f(\pi_i)=f(\pi_j)$,则称 f 是保序的.

DPA 和 IPA 算法都是针对静态优先级调度算法,例如 RM 和 DM(deadline monotonic)等算法可调度的任务集合进行优先级映射.DPA 算法按照保序规则,以优先级递减的顺序分配任务到系统优先级队列.映射后,同一队列内的任务按照 FIFO 方式执行,队列间任务仍然按原有静态优先级调度规则抢占执行.每分配一个任务,都要判定最大响应时间受影响任务的可调度性.如果这些任务可调度,则映射成立,否则将该任务分配到下一个较低的系统优先级队列中.映射过程中,如果在提供的系统优先级用完的情况下仍有未分配的任务,则说明优先级映射使系统变得不可调度.

IPA 算法按优先级递增的顺序将任务分配到系统优先级队列,映射不成立时将任务分配到下一较高的系统优先级.除此之外,其他遵循与 DPA 相同的规则.

1.2 TSM算法

利用抢占阈值调度模型,可实现一个双优先级系统.集合 Γ 中的任务 τ_i 不但具有任务优先级 $\pi_i\in[1,2,\ldots,n]$,还分配了抢占阈值 γ_i,并且 $\pi_i\le\gamma_i$.即 $\gamma_i\in[\pi_i,\ldots,n]$.任务优先级用于抢占其他任务,而抢占阈值是任务运行后的有效优先级.周期任务 τ_i 每次开始执行时,其优先级由 π_i 提升到 γ_i.执行完毕时,优先级由 γ_i 下降到 π_i.如果任务 τ_j 希望抢占当前正在运行的任务 τ_i,必须有 $\pi_j>\gamma_i$,抢占才能实现.

抢占调度和非抢占调度都是抢占阈值调度模型的特例.当任务的抢占阈值等于其任务优先级时,就是抢占调度;当任务的抢占阈值是集合中最高的任务优先级时,就是非抢占调度.因此,抢占阈值能同时利用抢占调度和非抢占调度的优点.通过调节任务的抢占阈值,能减少不必要的任务抢占,从而提高任务集合的可调度性.

在抢占阈值调度模型中,我们定义:

(1) 如果两个任务 τ_i 和 τ_j,存在 $\pi_i\le\gamma_j\wedge\pi_j\le\gamma_i$,那么这两个任务是相互非抢占的[9].

(2) 一组任务 $G=\{\tau_1,\tau_2,\ldots,\tau_l\}$,如果其中的任意两个任务 τ_i 和 τ_j 是相互非抢占的,那么这些任务构成了一个非抢占组(non-preemptive group,简称 NPG).

生成(或者称作分割)NPG 时,按照抢占阈值非递减的顺序,从具有最低抢占阈值的未分配任务开始,依次将相互非抢占的任务分配到同一 NPG 中.重复这一过程,直到分配完所有任务.对于其有相同抢占阈值的任务,按照优先级递增的顺序排列.假定分割结束后 NPG 的个数为 m,令 $G_i(1\le i\le m)$表示一个 NPG.从生成方法可以看出,用来构建 NPG 的第 1 个任务是关键任务,其抢占阈值决定了非抢占组中任务优先级的取值范围,我们称该任务为 NPG 的标志任务.G_i 的任意一个任务都能抢占 G_{i-1} 的标志任务,否则该任务会分配在 G_{i-1} 中.所以,G_i 中最小任务优先级一定大于 G_{i-1} 标志任务的抢占阈值.另外,一个 NPG 中的任务优先级是连续的,并且小于其标志任务的抢占阈值.

根据上述特点,我们能将一个 NPG 中所有任务优先级映射成一个系统优先级,在这一范围内的抢占阈值也映射成该系统优先级,这就是 TSM 算法.映射后,为了实现不同 NPG 中任务之间的抢占执行,我们将一个 NPG 对应一个线程,而 NPG 中的每个任务对应一个事件,事件到达就是相应任务就绪,线程处理事件就是在执行相应的任务.这里称作线程是为了与 NPG 中的任务相区分,在实际的实时操作系统(例如 VxWorks)中,一个线程仍然由一个任务来实现.映射后保留了任务优先级,用来排列组中任务的执行顺序.这就是事件驱动线程框架.因为线程框架和保留的任务优先级处于应用层,并不为底层实时操作系统所知,所以映射后,任务集合使用的系统优先级减少了,就是非抢占组的个数.

可以看出,抢占阈值调度中生成 NPG 的过程,就是在划分每个系统优先级所对应的任务优先级的范围,就是在执行优先级映射.因为这种映射没有引起新的阻塞时间,不影响任务的可调度性,所以不会产生优先级映射问题.虽然无须判定映射后任务的可调度性,但是为了比较,将抢占阈值调度模型使用的充要条件作为 TSM 算法的判定条件.

2 算法使用的判定条件

2.1 DPA算法和IPA算法使用的判定条件

令$\Gamma(higher,\tau_i)$表示映射后比τ_i系统优先级高的任务集合.$\Gamma(same,\tau_i)$表示映射后与任务τ_i具有相同系统优先级的任务集合.$\Gamma(same,other,\tau_i)$表示映射后与任务$\tau_i$具有相同系统优先级,但不包括$\tau_i$的任务集合.

文献[4]扩展文献[1]的判定条件,推导出在$D_i \le T_i$的情况下,能用于 DPA 和 IPA 算法的判定映射后任务τ_i可调度性的充要条件:

$$\min_{0\le t\le D_i}\left(W_i(t)/t\right)\le 1 \tag{1}$$

其中

$$W_i(t)=\sum_{\tau_q\in\Gamma(higher,\tau_i)}\left\lceil t/T_q\right\rceil\cdot C_q+\sum_{\tau_p\in\Gamma(same,\tau_i)}C_p \tag{2}$$

令 $k(k\ge 0)$为第一忙周期[11]中,任务τ_i释放的第 k 个请求的序号.$S_i(k)$表示任务τ_i的第 k 个请求到达的时间.令 l 为第一忙周期中任务τ_i释放的最后一个请求的序号,$W_i(l)$为第一忙周期的结束时刻.$W_i(l)-(l+1)T_i\le 0$ 时,第一忙周期结束.

文献[5]推导了在$D_i>T_i$的情况下,DPA 算法映射任务τ_i的优先级后,判定其可调度性的充要条件:

$$\max_{k=0,1,2,\ldots,l}W_i(t)\le D_i+k\cdot T_i \tag{3}$$

其中

$$W_i(l)-(l+1)\cdot T_i\le 0 \tag{4}$$

$$W_i(k)=(k+1)\cdot C_i+\sum_{\tau_q\in\Gamma(higher,\tau_i)}\left\lceil W_i(k)/T_q\right\rceil\cdot C_q+\sum_{\tau_p\in\Gamma(same,other,\tau_i)}\left(\left\lfloor S_i(k)/T_p\right\rfloor+1\right)\cdot C_p \tag{5}$$

$$S_i(k)=k\cdot T_i \tag{6}$$

式(3)~式(6)的应用环境与 IPA 算法的相同,所以能被后者用来判定$D_i>T_i$的情况下,映射后任务的可调度性.

2.2 TSM算法使用的判定条件

如前所述,令 TSM 算法使用的判定条件是抢占阈值调度模型中,分配任务优先级和抢占阈值,生成非抢占组时使用的判定条件[9].该条件没有限制任务的最终截止期,包含$D_i\le T_i$和$D_i>T_i$这两种情况.

在抢占阈值调度模型中,任务τ_i的最大阻塞时间是

$$B(\tau_i)=\max_j\{C_j::\gamma\ge\pi_i>\pi_j\} \tag{7}$$

抢占阈值调度扩展了 *level-i* 忙周期分析.如果用$E_i(k)(k\ge 1)$表示任务τ_i的第 k 个请求开始执行的时间,$F_i(k)$表示忙周期的结束时间,$(k-1)\cdot T_i$表示第 k 个请求的到达时间,则针对 $k=1,2,3,\ldots$,计算 $E_i(k)$和 $F_i(k)$,直到 $k=m$ 时,使得$F_i(k)\le k\cdot T_i$.这样,任务τ_i的最坏情况响应时间为

$$R_i=\max_{k=1,2,3,\ldots,m}(F_i(k)-(k-1)\cdot T_i) \tag{8}$$

其中

$$F_i(k)=E_i(k)+C_i+\sum_{\forall j,\pi_j>\gamma_i}\left(\left\lceil F_i(k)/T_j\right\rceil-\left(1+\left\lfloor E_i(k)/T_j\right\rfloor\right)\right)\cdot C_j \tag{9}$$

$$E_i(k)=B(\tau_i)+(k-1)\cdot C_i+\sum_{\forall j,\pi_j>\pi_i}\left(1+\left\lfloor E_i(k)/T_j\right\rfloor\right)\cdot C_j \tag{10}$$

如果$R_i\le D_i$,则任务τ_i是可调度的.

3 算法特性

3.1 DPA算法和IPA算法的特性

由前面的论述可以看出,DPA 和 IPA 是与 TSM 不同类型的映射算法.根据队列中任务的执行方式,我们称之为 FIFO 类保序映射算法.从后面的仿真实验将会看到,在保证系统可调度的前提下,TSM 算法使用了比 DPA 和 IPA 算法都少的系统优先级.因此修改文献[5]推导 DPA 算法特性时所描述的前提条件,得出:

(1) 对于给定的任务集合,在所有 FIFO 类保序映射算法中,DPA 算法所需优先级个数最少.

(2) 对于系统优先级个数已定的任务集合,如果存在其他 FIFO 类保序映射算法使得该任务集合可调度,则 DPA 算法也能使该集合可调度.

可以证明 IPA 算法具有同样的特性:

定理 **1**. 对于给定的任务集合,在所有 FIFO 类保序映射算法中,IPA 算法所需优先级个数最少.

证明:采用反证法.

假定任务集合Γ在 IPA 算法下可调度,需要的系统优先级个数是 m,1 为最低的系统优先级,m 为最高的系统优先级.令$\Gamma(l)(1\le l\le m)$表示 IPA 算法生成的系统优先级队列.如果存在另一 FIFO 类保序映射算法 A 也使Γ可调度,需要的任务优先级个数为 $m'(m'<m)$,则至少存在一个 IPA 算法生成的队列$\Gamma(l)(1\le l\le m)$,其任务在算法 A 下必须重新分配到其他队列中.如果将$\Gamma(l)$中的第 1 个任务分配到$\Gamma(l-1)$中,则会引起$\Gamma(l-1)$中至少一个任务不可调度,否则 IPA 算法会将该任务分配在$\Gamma(l-1)$中.如果将第 1 个任务分配到$\Gamma(l+1)$中,为了遵循保序映射规则,需要将$\Gamma(l)$中所有的任务都转移到$\Gamma(l+1)$中,这会引起$\Gamma(l)$中至少一个任务不可调度,否则 IPA 算法会将$\Gamma(l)$和$\Gamma(l+1)$中的任务分配在同一队列中.因此要将$\Gamma(l+1)$中的任务移至$\Gamma(l+2)$.以此类推,使得算法 A 不可能使用比 IPA 算法少的系统优先级,这与假设 $m'<m$ 相矛盾.所以 IPA 算法所需系统优先级个数最少. □

定理 **2**. 对于系统优先级个数已定的任务集合,如果存在其他 FIFO 类保序映射算法,使得该任务集合可调度,则 IPA 算法也能使该集合可调度.

证明:假定存在一个 FIFO 类保序映射算法使任务集合Γ可调度.需要的系统优先级个数是 m.令$\Gamma(l)(1\le l\le m)$表示该算法生成的系统优先级队列.将$\Gamma(2)$中具有最低任务优先级的任务 $\tau_{2,lowest}$ 转移到$\Gamma(1)$时,分析可调度性受影响的任务:

(1) $\Gamma(2)$中的任务.根据式(1)~式(6),$\Gamma(2)$中剩余任务减少了 $\tau_{2,lowest}$ 引起的 FIFO 阻塞时间,所以最大响应时间不会增加.

(2) $\Gamma(1)$中的任务.在移动 $\tau_{2,lowest}$ 之前,$\Gamma(1)$中的任务会被 $\tau_{2,lowest}$ 抢占.移动之后,$\Gamma(1)$中的任务会减去 $\tau_{2,lowest}$ 抢占引起的阻塞时间,增加 $\tau_{2,lowest}$ 引起的 FIFO 阻塞时间.根据式(1)~式(6),后者不可能大于前者.所以$\Gamma(1)$中原有任务的最大响应时间不会增加.

(3) 任务 $\tau_{2,lowest}$.根据式(1)~式(6),任务 $\tau_{2,lowest}$ 的 FIFO 阻塞时间增大,所以最大响应时间会增加,可能引起 $\tau_{2,lowest}$ 变得不可调度.但是,如果出现这种情况,IPA 算法会将 $\tau_{2,lowest}$ 分配在$\Gamma(2)$中.

如果 $\tau_{2,lowest}$ 在转移后仍然可调度,就能重复上述过程,将$\Gamma(2)$中下一个优先级最低的任务从$\Gamma(2)$转移到$\Gamma(1)$.如果针对$\Gamma(3)$到$\Gamma(m)$等队列继续这一过程,就能得到 IPA 算法映射的结果.也就是说,IPA 算法能使该任务集合可调度.定理成立. □

3.2 TSM算法的特性

进一步划分保序的定义.对于某一优先级映射算法 f,定义:

$\forall\pi_i\in\Pi,\forall\pi_j\in\Pi$,如果$\pi_i<\pi_j\Rightarrow f(\pi_i)<f(\pi_j)$或者 $f(\pi_i)=f(\pi_j)$,并且当 $f(\pi_i)=f(\pi_j)$时,不会引起新的阻塞时间,则称 f 是严格保序的.如果 $f(\pi_i)=f(\pi_j)$时,引起新的阻塞时间,则称 f 是部分保序的.

通过前面的分析可以看出,DPA 和 IPA 算法是部分保序的.在 TSM 算法下,对于Π中的任意两个任务 τ_i 和 τ_j,如果分配到相同的 NPG 中,则 $f_{TSM}(\pi_i)=f_{TSM}(\pi_j)$.因为两个任务是相互非抢占的,所以映射后不会引入新的阻塞时间.如果两个任务没有分配到同一 NPG,并且$\pi_i<\pi_j$,则必定有 $f_{TSM}(\pi_i)<f_{TSM}(\pi_j)$.所以 TSM 算法是严格保序的.

4 算法比较

DPA 和 IPA 算法将多个任务分配到一个 FIFO 队列,会引起优先级倒置,降低了任务集合的可调度性.如果系统提供的优先级较少,可能出现静态优先级调度算法下可调度的任务集合,优先级映射后变得不可调度.如果希望映射后能保持任务集合的可调度,就需要较多的系统优先级.

与 DPA 和 IPA 算法相比,TSM 算法利用抢占阈值调度模型,在提高任务集合可调度性的同时,使用更少的系统优先级.在文献[10]的仿真环境下,任务最大周期为 100 时,平均 100 个任务被分割成 14.3 个非抢占组.

4.1 计算复杂度的比较

从式(1)~式(6)可以看出,DPA 算法将任务 τ_i 分配到系统优先级队列时,会增加队列中原有任务的最大响应时间,需要重新判定这些任务的可调度性.而任务 τ_i 的最大响应时间不会增加,无须判定其可调度性.如果系统优先级个数为 m,任务个数为 n.将所有任务分配到一个队列时计算量最大,为 $1+2+\ldots+(n-1)=(n\cdot(n-1))/2$.将 n 个任务平均分配到 m 个队列时计算量最小,为 $((n/m)\cdot(n/m-1)\cdot m)/2=((n\cdot(n-m))/2m$.因此,DPA 算法的计算复杂度是 $O(n^2)$[5].为了减少优先级映射时的计算量,文献[5]提出了改进措施.对任务 τ_i,令

$$H_i=\min_{0\le t\le D_i}\left(t-\sum_{\tau_q\in\Gamma(higher,\tau_i)}\left\lceil t/T_q\right\rceil\cdot C_q\right) \tag{11}$$

将任务 τ_i 分配到系统优先级为 $l(\tau_i)$的队列时,令 $H=\min_{l(\tau_j)=l(\tau_i)}(H_j)$.如果 $H<\sum_{\tau_p\in\Gamma(\text{same},\tau_i)}C_p$,必须将 τ_i 分配到下一个较低的优先级队列,否则分配成立,重新计算本队列的 H 值.此时,每分配一个任务,只需 1 次比较计算和 1 次生成最新 H 值的计算,所以计算复杂度是 $O(n)$.

从定理 2 的推导过程可以看出,IPA 算法将任务 τ_i 分配到系统优先级队列时,队列中原有任务的最大响应时间不会增加,而 τ_i 的最大响应时间会增大,因此只需判定 τ_i 是否可调度.所以,IPA 算法的计算复杂度是 $O(n)$.TSM 算法映射优先级时没有引起新的阻塞时间,所以映射后无须判定任务的可调度性,也就没有因此造成的计算量.

4.2 性能比较

4.2.1 性能指标

优先级映射算法的理想目标是:在不影响任务集合可调度性的前提下,使用尽可能少的系统优先级.所以,我们将算法在保证任务集合可调度的前提下,映射后所用优先级个数作为指标,比较 3 种算法的性能.具体包括算法在每个测量点所用优先级的最大值(max)、最小值(min)和平均值(ave).令 DPA 和 IPA 算法对 DM 算法可调度的任务集合进行优先级映射.

4.2.2 仿真条件

使用随机产生的任务集合.生成任务时,有两个参数是变化的:(1) 任务个数 totalTasks,从 5 开始,以 5 为步长递增到 50;(2) 任务集合的最大周期 maxPeriod,取 100 和 1000 两个值.对任意一组任务个数和最大周期,任务集合按如下规则产生:

(1) 在[1,maxPeriod]之间均匀、随机地选择任务周期 T_i.

(2) 在[0.1/totoalTasks,2.0/totalTask]之间均匀、随机地选择任务利用率 U_i.任务执行时间 $C_i=U_i\times T_i$.这里用任务个数来调整取值,以免产生过多的不可调度任务集合.

(3) 任务截止期 $D_i=T_i$.

针对任务个数和最大周期的每一组取值,从 100 次独立仿真实验中获得算法所用优先级的最大值、最小值和平均值.每次实验生成的任务集合既是 DM 算法可调度,也是抢占阈值调度模型可调度的.

4.2.3 结果分析

(1) maxPeriod=100 时的比较结果

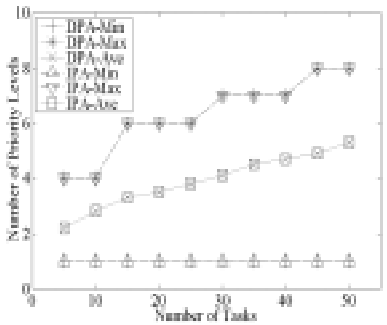
图 1(a)给出 maxPeriod=100 时 DPA 算法和 IPA 算法的性能比较.可以看出,在每个测量点,DPA 和 IPA 算法

得到了相同的最大值、最小值和平均值.说明两种算法需要相同个数的系统优先级.图 1(b)给出了 maxPeriod=100 时 DPA 与 TSM 算法的性能比较.可以看出,TSM 算法得到的最小值与 DPA 算法相同,但最大值要小,并且随着任务个数的增加,差值越来越大.当任务集合有 50 个任务时,TSM 算法得到的最大值比 DPA 算法小 5.TSM 算法得到的平均值随着任务集合的增加呈递减趋势,而 DPA 算法则呈递增趋势.任务集合包含 50 个任务时,两者之间的差值达到 5.

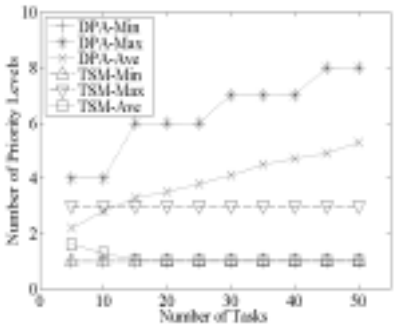
这些数据表明,虽然 3 种算法在每个测量点得到相同的最小值.但在大部分情况下,DPA 和 IPA 算法会使用比 TSM 算法更多的系统优先级.

(2) maxPeriod=1000 时的比较结果

图 2(a)给出 maxPeriod=1000 时 DPA 与 IPA 算法的性能比较.两种算法在每个测量点使用了相同的系统优先级.图 2(b)给出 IPA 和 TSM 算法的性能比较.可以看出,与 maxPeriod=100 时 DPA 与 TSM 算法的比较结果相似.所以 DPA 算法和 IPA 算法的性能相同,并且在大部分情况下,会使用比 TSM 算法更多的系统优先级.

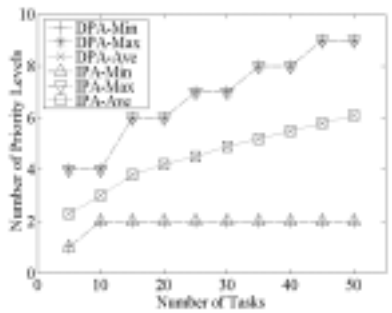


(a) Number of priority levels used by DPA as compared to IPA, with maxPeriod=100
(a) maxPeriod=100 时 DPA 算法和 IPA 算法使用的系统优先级个数

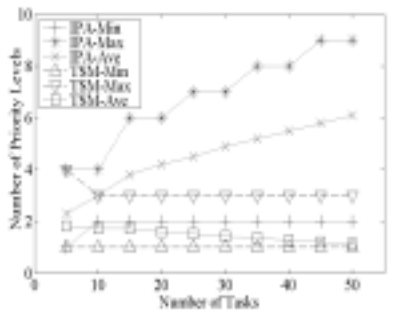


(b) Number of priority levels used by DPA as compared to TSM, with maxPeriod=100
(b) maxPeriod=100 时 DPA 算法和 TSM 算法使用的系统优先级个数

Fig.1
图 1



(a) Number of priority levels used by DPA as compared to IPA, with maxPeriod=1000
(a) maxPeriod=1000 时 DPA 算法和 IPA 算法使用的系统优先级个数



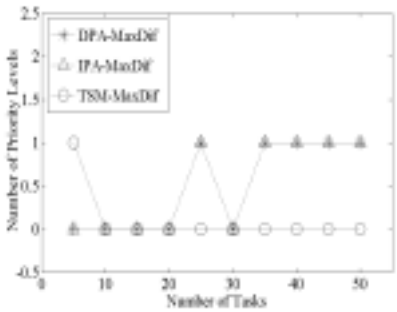
(b) Number of priority levels used by IPA as compared to TSM, with maxPeriod=1000
(b) maxPeriod=1000 时 IPA 算法和 TSM 算法使用的系统优先级个数

Fig.2
图 2

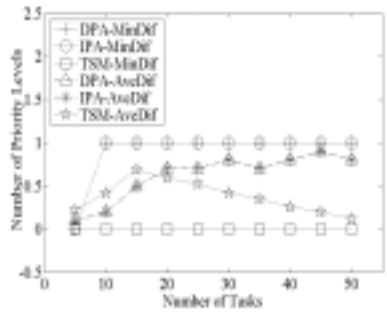
(3) 比较 maxPeriod 不同取值时,各个值的变化

maxPeriod=1000 时,每个测量点算法得到的最大值、最小值和平均值大于或者等于 maxPeriod=100 时得到的值.用 MaxDif,MinDif 和 AveDif 表示这一差值.图 3(a)给出 3 种算法在每个测量点的 MaxDif 值.可以看出,DPA 和 IPA 算法的 MaxDif 值呈递增趋势,并且大于或者等于 TSM 算法的取值.后者呈递减趋势,任务个数到 10 以后,算法所用优先级个数的最大值没有变化.这说明,与 TSM 算法相比,DPA 和 IPA 算法所用系统优先级个数的最大

值更易受 maxPeriod 取值的影响.图 3(b)给出 3 种算法在每个测量点的 MinDif 和 AveDif 值.可以看出, DPA 和 IPA 算法的 MinDif 值呈递增趋势,而且大于或者等于 TSM 算法的取值.后者一直为 0.这同样说明,DPA 和 IPA 算法所用系统优先级个数的最小值更易受 maxPeriod 取值的影响.在任务集合有 20 个任务之前,TSM 算法的 AveDif 值大于 DPA 和 IPA 算法.但是集合有 20 个任务之后,DPA 和 IPA 算法的 AveDif 值要大于 TSM 算法,呈递增趋势,而后者呈递减趋势,造成两者之间的差值越来越大.这也说明,与 TSM 算法相比,在大部分情况下,DPA 和 IPA 算法在每个测量点所用系统优先级的平均值,会随着 maxPeriod 取值的增大,有更多的增加.总之,在相同条件下,针对同一任务集合,DPA 和 IPA 算法会使用比 TSM 算法更多的系统优先级.当 maxPeriod 取值增大时,DPA 和 IPA 算法所用系统优先级的增幅要大于 TSM 算法.



(a) MaxDif value of algorithms
(a) 3 种算法的 MaxDif 值



(b) MinDif and AveDif values of algorithms
(b) 3 种算法的 MinDif 和 AveDif 值

Fig.3
图 3

文献[5]中通过仿真实验说明了 DPA 算法优于文献[3]提出的常量法,与后者相比能获得较大的任务集合可调度率.根据前面的论述以及定理 1 和定理 2,可以得出 IPA 算法与 DPA 算法具有相同的性能,同样优于常量法.

4.3 算法的适用环境

根据不同的特点和性能,得出算法的不同适用环境.如果任务个数与系统提供的优先级个数接近,而且任务集合是静态优先级调度算法可调度的,可以尝试用 DPA 或 IPA 算法进行优先级映射,这样实现简单.例如,在使用 VxWorks 的系统中,如果系统提供的优先级个数与任务个数接近,并且映射后任务集合可调度,则使用 DPA 或 IPA 算法.映射后,无须更改任务源程序,只需改变其优先级.具有相同系统优先级的任务利用 VxWorks 提供的 FIFO 调度机制执行,不同系统优先级的任务仍然使用 VxWorks 提供的抢占式调度机制执行.如果与上述情况相反,则尝试用 TSM 算法进行优先级映射,并在应用层实现事件驱动线程框架.例如,上述使用 VxWorks 的实时系统,如果任务个数远大于系统提供的优先级个数,当使用 DPA 或者 IPA 算法时,映射后任务集合不可调度,则使用 TSM 算法映射任务优先级.但此时需要改变相关任务的源程序,将一个 NPG 中的多个任务由一个线程(VxWorks 的任务)来处理,任务程序对应线程中的事件处理程序.线程之间利用 VxWorks 提供的抢占式调度机制执行,线程负责排列内部事件的执行顺序.虽然增加了实现应用程序的复杂度,但能提高任务集合的可调度性,并能减少所用系统优先级个数.

5 结 语

通常情况下,实时嵌入式系统提供的优先级个数是有限的.当任务个数多于系统提供的优先级个数时,就可能引起优先级映射问题.当前主要解决方案有 DPA,IPA 以及本文提出的 TSM 映射算法.DPA 和 IPA 算法属于 FIFO 类保序映射算法,只能尽力查找保持系统可调度性的映射方式,不能从根本上解决优先级映射问题.如果系统提供的优先级过少,就可能出现映射后的任务集合一定不可调度的情况.而 TSM 算法没有引起新的阻塞时间,所以映射后不会产生优先级映射问题.同时,抢占阈值调度模型能够提高任务集合的可调度性.TSM 算法使用较少系统优先级,但需要事件驱动线程框架的配合.而 DPA 和 IPA 算法不需要任何应用框架的支持,映射后的

任务集合可以直接运行在实时操作系统上.

致谢 在此,我们向对本文提出宝贵意见的评审老师表示感谢.

References:

[1] Lehoczky JP, Sha L, Ding Y. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In: Proc. of the Real-Time Systems Symp. Santa Monica: IEEE Computer Society Press, 1989. 166−171.

[2] Liu CL, Layland JW. Scheduling algorithms for multiprogramming in a hard-real-time environment. Journal of the ACM, 1973,20(1):46−61.

[3] Lehoczky JP, Sha L. Performance of real-time bus scheduling algorithms. ACM SIGMETRICS Performance Evaluation Review, 1986,14(1):44−53.

[4] Katcher DI, Sathaye SS, Strosnider JK. Fixed priority scheduling with limited priority levels. IEEE Trans. on Computers, 1995,44(9):1140−1144.

[5] Bin XL, Yang YH, Jin SY. Optimal fixed priority assignment with limited priority levels. LNCS 2834, Springer-Verlag, 2003. 194−203.

[6] Orozco J, Cayssials R, Santos J, Santos R. On the minimum number of priority levels required for the rate monotonic scheduling of real-time systems. In: Proc. of the 10th EUROMICRO Workshop on Real Time Systems. 1998. http://www.mrtc.mdh.se/emrt98/wip/proceedings/5.ps

[7] Guo CG, Wang HM, Zou P, Wang F. Priority mapping in real-time middleware. Journal of Software, 2003,14(6):1060−1065 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/1060.html

[8] Dpippo LC, Wolfe VF, Esibov L, Cooper G, Bethmangalkar R, Johnston R, Thuraisingham BM, Mauer J. Scheduling and priority mapping for static real-time middleware. Real-Time Systems, 2001,20(2):155−182.

[9] Wang Y, Saksena M. Scheduling fixed-priority tasks with preemption threshold. In: Gakkai JS, ed. Proc. of the 6th Int'l Conf. on Real-Time Computing Systems and Application. Los Alamitos: IEEE Computer Society, 1999. 328−335.

[10] Saksena M, Wang Y. Scalable real-time system design using preemption threshold. In: Jeffay K, ed. Proc. of the 21st IEEE Real-Time Systems Symp. Los Alamitos: IEEE Computer Society Press, 2000. 25−34.

[11] Lehoczky JP. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In: Proc. of the 11th Real-Time Systems Symp. IEEE Computer Society Press, 1990. 201−213.

附中文参考文献:

[7] 郭长国,王怀民,邹鹏,王峰.实时中间件的优先级映射.软件学报,2003,14(6):1060−1065. http://www.jos.org.cn/1000-9825/14/1060 html.

王保进(1974－),男,河北保定人,博士生,主要研究领域为嵌入式系统,实时系统,通信技术.

王志刚(1944－),男,教授,博士生导师,主要研究领域为无线电通信技术,信号处理,实时系统.

李明树(1966－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为需求工程,软件过程,实时系统.