计算机研究与发展
Journal of Computer Research and Development

ISSN 1000-1239/CN 11-1777/TP
43(6): 1083~1089, 2006

一种不影响任务集合可调度性的优先级映射算法

王保进[1,2]　　李明树[2,3]
[1](解放军信息工程大学信息工程学院　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)
[3](中国科学院软件研究所计算机科学重点实验室　北京　100080)
(baojin-wang@itechs.iscas.ac.cn)

A Priority Mapping Algorithm Without Affecting the Schedulability of Tasks Set

Wang Baojin[1,2] and Li Mingshu[2,3]
[1](*Institute of Information and Engineering, PLA Information and Engineering University, Zhengzhou* 450002)
[2](*Laboratory for Internet Software Technologies, Institute of Software, Chinese Academy of Sciences, Beijing* 100080)
[3](*Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing* 100080)

Abstract　In practice, the schedulability of static priority scheduling may be reduced if priority levels of the system are insufficient. When the priority levels that a task set requires are more than the system can support, more than one task must be assigned the same priority. The priority mapping algorithms that have been used can augment the worst-case response time of task with higher priority, and may reduce the system schedulability. If keeping system schedulable, these algorithms may require more priority levels than the system can support. With preemption threshold scheduling model, the schedulability is improved as compared to both preemption and non-preemption scheduling models, and all tasks can be handled with the minimum number of event handling threads. But, the number of system priority levels that a thread needs does not be reduced. This paper presents a priority mapping algorithm called TSM (threshold segment mapping) and a thread implementation architecture with event-driven. It is shown that the TSM algorithm can maintain the strict order of tasks' priorities, and does not improve task's worst-case response time. The number of system priority levels used by the algorithm and architecture equals to that of threads. Simulations show that when maintaining system schedulability, the TSM algorithm uses less priority levels than existing priority mapping algorithms.

Key words　priority mapping; preemption threshold; non-preemptive group; event-driven; strict order

摘　要　采用静态优先级调度的实时系统中,当任务个数多于优先级个数时,只能给多个任务分配相同的优先级.现有分配算法增大了高优先级任务的最坏情况响应时间,可能造成任务集合不可调度.利用抢占阈值的调度算法,能在提高任务集合可调度性的同时,使用较少的线程.但所用优先级个数没有减少.提出了一种优先级映射算法——阈值段间映射法(threshold segment mapping, TSM),以及与之配合的事件驱动线程框架.证明了TSM是严格排序的.仿真结果表明,在保证任务集合可调度的前提下,TSM使用了比现有映射算法更少的优先级.

关键词　优先级映射;抢占阈值;非抢占组;事件驱动;严格排序

中图法分类号　TP311

收稿日期:2005-03-25;修回日期:2005-11-24
基金项目:国家"八六三"高技术研究发展计划基金项目(2004AA1Z2050)

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

1 引 言

现有的静态(固定)优先级调度算法[1,2]都假设系统提供无限多的优先级,而实际中,系统优先级个数是有限的. 例如,VxWorks 提供 256 个优先级,实时 POSIX 支持 32 个优先级. 当任务个数多于系统支持的优先级个数时,只能给几个任务分配相同的系统优先级. 这需要一种优先级映射算法.

目前,处理优先级有限的方法分为 3 类:①给出映射方法,未提供判定公式[3]. ②给出判定公式,未提供映射方法[4,5]. ③既给出映射方法,又提供判定公式. 包括文献[6]提出的按照优先级递增的顺序依次分配任务到系统优先级组的映射方法,本文称做 IPA(increasing priority assignment)算法;文献[7]提出的按照优先级递减的顺序依次分配任务到系统优先级组的方法,本文称做 DPA(decreasing priority assignment)算法.

映射算法应该给出映射方法并提供判定公式. 因为判定公式能证明映射方法的有效性,而映射方法能实现优先级映射. 因此,本文将上述第 3 类方法称做优先级映射算法. 本文提出了一种优先级映射算法——阈值段间映射法(TSM),与抢占阈值调度模型结合,利用其公式判定任务集合的可调度性,推导的映射方法与事件驱动线程框架配合,将所用系统优先级减少到系统线程的个数. 并通过仿真实验,比较分析了 TSM, IPA 和 DPA 算法的性能. 结果表明,与另外两种算法相比,TSM 算法能在提高任务集合可调度性的同时,使用更少的系统优先级.

2 优先级映射问题和抢占阈值

定义 $S=\{\tau_1, \tau_2, \cdots, \tau_n\}$ 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 $\tau_i=(T_i, C_i, D_i)$, $i=1, 2, \cdots, n$ 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. 对于静态优先级调度算法,如果任务集合包含 n 个任务,就会分配 n 个任务优先级. 我们假定大的数值表示高的优先级,对于集合 S,最小的优先级是 1,最大的优先级是 n. 在下面的论述中,系统优先级又称做线程优先级,优先级映射又称做将任务分配到系统优先级组中,系统优先级组又称做 FIFO 队列.

2.1 优先级映射问题

静态优先级调度在系统支持的优先级个数有限时,要解决优先级映射问题,包括:①如何将多个不同的任务优先级映射成一个系统优先级. ②优先级映射后如何判定任务集合的可调度性. 现有的映射算法[6,7]针对静态优先级调度算法(例如:RM, DM)可调度的任务集合进行优先级再分配,按照递增或者递减的优先级顺序将任务分配到系统优先级组中. 每映射一个优先级,都要进行可调度性判定. 文献[5~7]中的公式都是对文献[1]中判定公式的扩展,均假定映射后,组间的任务按照静态优先级调度算法调度执行,组内的任务按照 FIFO 方式执行. 这会造成同一 FIFO 队列中,具有较高任务优先级的任务被排在队列前面的具有较低任务优先级的任务阻塞,从而引起优先级倒置,增大高优先级任务的最坏情况响应时间,降低任务的可调度性. 如果希望映射后仍能保持集合的可调度性,就会需要较多的系统优先级. 因此,现有的映射算法很难在保持任务集合可调度的同时使用较少的系统优先级.

系统需要一种能严格排序的映射算法. 即映射不改变任务的优先级顺序,不引入新的阻塞时间. 下面给出严格排序的定义.

定义 1. 严格排序.

任务优先级用 Π 表示: $\Pi=\{\pi_1, \pi_2, \cdots, \pi_n\}$, $\forall\pi_i\in\Pi(1\leqslant i\leqslant n)$, $\forall\pi_j\in\Pi(1\leqslant j\leqslant n)$,如果 $i\neq j$,则 $\pi_i\neq\pi_j$. 线程优先级用 X 表示: $X=\{\chi_1, \chi_2, \cdots, \chi_m\}$, $\forall\chi_i\in X(1\leqslant i\leqslant m)$, $\forall\chi_j\in X(1\leqslant j\leqslant m)$,如果 $i\neq j$,则 $\chi_i\neq\chi_j$. 这里 $n\geqslant m\geqslant 1$,我们也用 x 表示 Π 中的元素. 对于某一优先级映射算法 f,我们定义: $\forall\pi_i\in\Pi$, $\forall\pi_j\in\Pi$,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 是严格排序的.

现有的 IPA 和 DPA 两种算法虽然在映射后保持了任务的优先级顺序,但引入新的阻塞时间,所以不是严格排序的映射算法.

2.2 抢占阈值

利用抢占阈值的调度模型,在实际上实现了一个双优先级系统. 集合 S 中的任务 τ_i 不但具有任务优先级 $\pi_i\in[1, 2, \cdots, n]$,还分配了抢占阈值 γ_i,并且 $\pi_i\leqslant\gamma_i$. 即, $\gamma_i\in[\pi_i, \cdots, n]$. 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,那么必须有

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

$\pi_j > \gamma_i$, 抢占才能实现.

当每个任务的抢占阈值等于其任务优先级时, 就是抢占调度. 当每个任务的抢占阈值是系统最高的优先级时, 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, 减少不必要的任务抢占, 来提高任务集合的可调度性. 抢占阈值调度能将任务集合分割成少量的非抢占组(non-preemptive group, NPG), 其定义如下:

(1) 如果两个任务 τ_i 和 τ_j, 存在 $\pi_i \leqslant \gamma_j \wedge \pi_j \leqslant \gamma_i$, 那么这两个任务是相互非抢占的[8].

(2) 一组任务 $G = \{\tau_1, \tau_2, \cdots, \tau_l\}$, 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.

使用抢占阈值时, 系统开发过程如下:

(1) 分配任务的优先级和抢占阈值, 并进行可调度性判定.

(2) 将任务集合分割成最小数目的 NPG.

(3) 利用事件处理线程框架, 将一个 NPG 对应一个线程, 并被实时操作系统调度执行. 线程中的每个事件对应 NPG 的一个任务. 事件到达就意味着相应任务一个新周期的开始. 处理事件就是在处理相应的任务.

利用抢占阈值, 不但能提高实时嵌入式系统的可调度性, 还能减小并发线程的个数. 在文献[8]的仿真环境下, 当任务最大周期为 100 时, 平均 100 个任务被分割成 14.3 个 NPG.

3 TSM 算法

3.1 分割 NPG 的方法

将任务集合分割成若干 NPG 的过程如下:

(1) 按照抢占阈值非递减的顺序排列任务. 对于具有相同抢占阈值的任务, 按照优先级递增的顺序排列. 并用 L 表示结果列表.

(2) 在 L 中选择抢占阈值最低的任务 τ_i, 构建一个非抢占组 G, 如果存在多个具有相同最低抢占阈值的任务, 那么选择优先级最高者. 对于 L 中的其他任务 τ_k, 如果 $\pi_k \leqslant \gamma_i$, 就将其从 L 中分离出来, 并分配到 G 中. 因为 L 按照非递减的抢占阈值排列, 所以必定存在 $\pi_i \leqslant \gamma_k$. 在检查完 L 中所有任务后, 就生成了一个 NPG.

(3) 重复过程(2), 直到分割完 L 中所有任务.

已证明, 该分割算法能保证 NPG 中任务之间一定是非抢占的, 而且算法是最优的[8].

3.2 分割方法的特点

假定分割结束后 NPG 的个数为 m. 令 G_i ($1 \leqslant i \leqslant m$) 表示一个 NPG. 分析上面的分割步骤可以看出, 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. 称该任务为 NPG 的标志任务, 用 R_i ($1 \leqslant i \leqslant m$) 表示, π_{R_i} ($1 \leqslant i \leqslant m$) 是其优先级值, γ_{R_i} ($1 \leqslant i \leqslant m$) 是其抢占阈值. 又令 $Min_{G_i} = \min\limits_{\tau_j \in G_i} \pi_j$. 非抢占组存在下面的特性:

引理 1. 对于集合 S 的任意一个任务 τ_j, 如果其优先级 $\pi_j \leqslant \gamma_{R_i}$ ($1 \leqslant i \leqslant m$), 那么该任务必定存在于 NPG 集合 $\{G_k \mid 1 \leqslant k \leqslant i\}$ 的某一元素中.

证明.

(1) 如果任务 τ_j 的抢占阈值 $\gamma_j \geqslant \gamma_{R_i}$, 因为 $\gamma_{R_i} \geqslant \pi_{R_i}$, 所以 $\gamma_j \geqslant \pi_{R_i}$. 根据生成 NPG 的方法, 该任务一定和 R_i 一起存在于 G_i 中.

(2) 如果 $\gamma_j < \gamma_{R_i}$, 因为生成 NPG 时, 按照抢占阈值非递减的顺序排列任务, 如果 $\tau_j \notin G_i$, 那么, $\exists G_k (k < i)$, 使得 $\tau_j \in G_k$. 证毕.

引理 2. 对于任意相邻的两个非抢占组 G_i 和 G_{i-1} ($2 \leqslant i \leqslant m$), G_i 中的最小任务优先级一定大于 G_{i-1} 标志任务的抢占阈值. 即, $Min_{G_i} > \gamma_{R_{i-1}}$.

证明. 根据生成 NPG 的方法, 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. 因此引理 2 成立. 证毕.

定理 1. 一定有 $G_1 = \{\tau_k \mid 1 \leqslant \pi_k \leqslant \gamma_{R_1}\}$, $G_i = \{\tau_k \mid \gamma_{R_{i-1}} + 1 \leqslant \pi_k \leqslant \gamma_{R_i}, 2 \leqslant i \leqslant m\}$.

证明.

(1) 根据引理 1, 如果任务 τ_k 的优先级 $1 \leqslant \pi_k \leqslant \gamma_{R_1}$, 那么一定有 $\tau_k \in G_1$. 因此 $G_1 = \{\tau_k \mid 1 \leqslant \pi_k \leqslant \gamma_{R_1}\}$.

(2) 根据引理 1 和引理 2, 如果任务 τ_k 的优先级 $\gamma_{R_{i-1}} + 1 \leqslant \pi_k \leqslant \gamma_{R_i}$ ($2 \leqslant i \leqslant m$), 那么一定有 $\tau_k \in G_i$. 所以 $G_i = \{\tau_k \mid \gamma_{R_{i-1}} + 1 \leqslant \pi_k \leqslant \gamma_{R_i}, 2 \leqslant i \leqslant m\}$.

证毕.

3.3 TSM 算法

我们知道, 如果线程之间具有不同的优先级, 就能实现抢占调度. 从定理 1 可以看出, 如果将任务集合的优先级按顺序排列, 构成一条"线段", 那么一个 NPG 中所有任务的优先级将对应其中连续的一段. 如果有 m 个 NPG, 这条线段就会有互不相交的 m 段. 如果将每一段映射成一个系统优先级, 则会产生 m 个不同的系统优先级. 抢占阈值根据所在

?1994-2013 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

区间,进行相同规则的映射,也会产生 m 个不同的映射后抢占阈值. 如果将系统优先级作为就绪线程抢占当前运行线程所用的优先级,将映射后抢占阈值作为线程运行时的优先级,那么进行上述映射后仍然能实现与映射前相同的线程间抢占. 这就是阈值段间映射法(TSM). TSM 算法所用的系统优先级个数就是任务集合分割成 NPG 的个数.

图 1 表示了映射规则,其中的 $P(i)=\gamma_{R_i}(1\leqslant i\leqslant m)$,假定线程优先级从 1 递增到 m.

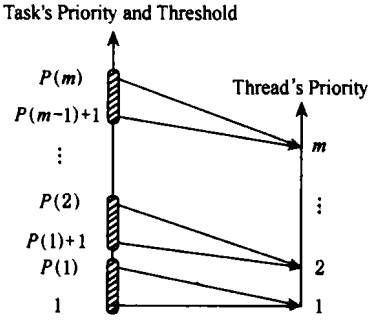

Fig. 1 Threshold segment mapping.

图 1 阈值段间映射法

优先级映射算法 f_{TSM} 是定义在 Π 上的函数. 即, $f_{TSM}: \Pi\rightarrow X$.

$$f_{TSM}(x)=\begin{cases}\chi_1, & 1\leqslant x\leqslant \gamma_{R_1},\\ \chi_i, & \gamma_{R_{i-1}}< x\leqslant \gamma_{R_i}(2\leqslant i\leqslant m).\end{cases}$$

定理 2. f_{TSM}是严格排序的.

证明. 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 f_{TSM}是严格排序的. 证毕.

TSM 算法也包括了判定任务集合可调度性的公式,就是抢占阈值调度模型的判定公式[9]. 因为 TSM 算法没有引入新阻塞时间,不影响任务集合的可调度性,所以映射优先级后,无需重新判定任务是否可调度. IPA 和 DPA 算法引入了新阻塞时间,所以系统开发过程中,既需要在映射前判定集合的可调度性,也需要在映射后重新判定任务的可调度性. TSM 算法中,任务 τ_i 的线程优先级也称做映射后任务优先级,用 υ_i 表示,映射后抢占阈值用 ω_i 表示.

4 事件驱动线程框架

在文献[8]中,每个 NPG 由一个事件处理线程来实现,每个线程维护一个事件队列,队列中事件的优先级和抢占阈值就是相应任务的优先级和抢占阈值. 对于就绪的线程,其优先级是队列中最高的事件优先级,每次事件到达后,都以此优先级来试图抢占当前正在运行的线程. 当抢占成功,线程被操作系统调度执行后,其优先级重置为队列中最高优先级事件的抢占阈值. 当线程处理完一个事件后,会将其从队列中清除. 此时,线程优先级又被重置为当前队列中最高的事件优先级,并以此与其他线程竞争处理器.

可以看出,这种线程框架虽然减少了调度实体(线程)的数目,但使用任务映射前的参数对(π_i, γ_i)时,没有减少所用系统优先级的个数,使用 TSM 算法映射后的参数对(υ_i, ω_i)时,无法排列线程内事件的执行顺序. 为了解决这些问题,提出了事件驱动线程框架. 每个事件 E_i 都有一个三元组参数(π_i, υ_i, ω_i). 其中的任务优先级 π_i 用于队列内事件排序,线程优先级 υ_i 和映射后抢占阈值 ω_i 用于实时操作系统的抢占调度. 事件驱动线程框架遵循如下原则.

原则 1. 每个线程维护一个事件队列,并按照递减的 π_i 值排列未处理事件.

原则 2. 队列中没有事件的线程处于阻塞状态,线程优先级设置为 0. 至少有一个事件到达后,线程进入就绪态. 用 υ 表示队列中最高优先级事件的映射后优先级. 有事件到达后,就绪态线程的优先级被重置为当前线程优先级与 υ 之间的最大值,并以此优先级试着抢占当前正在运行的线程.

原则 3. 线程获得处理器使用权后,会将其优先级重置为最高优先级事件的映射后抢占阈值. 正在运行的线程总是选择队列中优先级最高的事件来处理,在处理时不会被本队列中后续到达的事件抢占. 处理完一个事件后,线程优先级又被重置为当前队列中最高优先级事件的映射后优先级. 并以此与其他线程竞争处理器. 运行线程能被其他高优先级线程抢占. 对于被抢占的线程,即使有更高优先级的后续事件到达,其优先级仍然保持不变.

5 算法分析

后面的第 5.1 节以公开发表的文献[10]中使用的任务集合作为具体实例,描述 TSM 算法的使用过程. 第 5.2 节进行仿真实验,比较 TSM 算法与现有映射算法的性能.

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

5.1 实例分析

Olympus 人造卫星的姿态和轨道控制系统包含 21 个任务(包括周期任务和偶发性任务). 表 1 列出了任务周期和最坏情况执行时间. 因为文献没有给出任务截止期，我们假设截止期等于任务周期. 利用文献[8]的算法可以得到任务的优先级和抢占阈值，如表 1 所示：

Table 1 Tasks List

表 1 任务列表

Task	Period	WCET	Priority	Threshold	Task	Period	WCET	Priority	Threshold
1	100	4.08	20	21	12	1000	83.02	13	19
2	1000	2.06	11	21	13	100	24.62	21	21
3	500	4.12	18	21	14	1000	63.7	14	21
4	2000	8.25	4	21	15	500	5.32	19	21
5	625	2.06	16	21	16	2000	32.02	8	21
6	1870	2.06	5	21	17	1000	22.52	15	21
7	1000	2.06	12	21	18	2000	32.02	9	21
8	10000	99.32	2	19	19	1870	51.5	10	21
9	2000	45.82	6	21	20	625	49.8	17	21
10	2000	428.82	7	10	21	36000	9.42	1	21
11	10000	58.52	3	21					

利用分割算法对任务分组，得到 3 个 NPG，如表 2 所示. 这里用参数对 (π_i, γ_i) 表示任务 τ_i. G_1 的标志任务是 τ_{10}，G_2 的标志任务是 τ_{12}，G_3 的标志任务是 τ_{13}. 3 个任务的抢占阈值分别是 10，19 和 21. 可以看出，G_1 包含了优先级从 1 到 10 的任务，G_2 包含了优先级从 11 到 19 的任务，G_3 包含了优先级从 20 到 21 的任务，符合定理 1 的结论. 因此，标志任务的抢占阈值就是映射时优先级分段的界限值

Table 2 Non-Preemption Groups List

表 2 NPG 组列表

Before Mapping		After Mapping	
NPG	Tasks	NPG	Tasks
G_1	$\tau_{21}(1,21)$, $\tau_8(2,19)$, $\tau_{11}(3,21)$, $\tau_4(4,21)$, $\tau_6(5,21)$, $\tau_9(6,21)$, $\tau_{10}(\mathbf{7,10})$, $\tau_{16}(8,21)$, $\tau_{18}(9,21)$, $\tau_{19}(10,21)$	G_1	$\tau_{21}(1,3)$, $\tau_8(1,2)$, $\tau_{11}(1,3)$, $\tau_4(1,3)$, $\tau_6(1,3)$, $\tau_9(1,3)$, $\tau_{10}(\mathbf{1,1})$, $\tau_{16}(1,3)$, $\tau_{18}(1,3)$, $\tau_{19}(1,3)$
G_2	$\tau_2(11,21)$, $\tau_7(12,21)$, $\tau_{12}(\mathbf{13,19})$, $\tau_{14}(14,21)$, $\tau_{17}(15,21)$, $\tau_5(16,21)$, $\tau_{20}(17,21)$, $\tau_3(18,21)$, $\tau_{15}(19,21)$	G_2	$\tau_2(2,3)$, $\tau_7(2,3)$, $\tau_{12}(\mathbf{2,2})$, $\tau_{14}(2,3)$, $\tau_{17}(2,3)$, $\tau_5(2,3)$, $\tau_{20}(2,3)$, $\tau_3(2,3)$, $\tau_{15}(2,3)$
G_3	$\tau_1(20,21)$, $\tau_{13}(\mathbf{21,21})$	G_3	$\tau_1(3,3)$, $\tau_{13}(\mathbf{3,3})$

之后，利用 TSM 算法映射优先级. 表 2 右侧列出了映射后任务优先级和抢占阈值，此时的任务 τ_i 用参数对 (υ_i, ω_i) 表示

5.2 性能比较

5.2.1 性能指标

在保证任务集合可调度的前提下，映射后所用的系统优先级个数作为性能指标，将本文提出的 TSM 算法与现有优先级映射算法进行比较，包括最大值(max)、最小值(min)和平均值(ave). 这里令 IPA 和 DPA 算法对 DM 算法可调度的任务集合进行优先级映射.

5.2.2 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，任务集合按下面的规则产生：

(1) 在 $[1, maxPeriod]$ 之间均匀、随机地选择任务周期 T_i.

(2) 在 $[0.1/totalTasks, 2.0/totalTask]$ 之间均

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

匀、随机地选择任务利用率 U_i. 任务的执行时间 $C_i = U_i \times T_i$. 这里用任务个数来调整数值, 是为了避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每个算法的3个性能指标值.

5.2.3 结果分析

(1) $maxPeriod = 100$ 时的结果

图2(a)和图2(b)分别给出 $maxPeriod = 100$ 时TSM算法与IPA算法和DPA算法的性能比较. 可以看出, 在每个测量点, IPA和DPA算法得到的最小值、最大值以及平均值相同. 说明两个算法需要相同的系统优先级. TSM算法得到的最小值与其他两个算法相同, 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其他两个算法小5. 由TSM算法得到的平均值随着任务集合的增加呈递减趋势, 而其他两个算法则呈递增趋势. 有50个任务时, 相互之间的差值达到5.

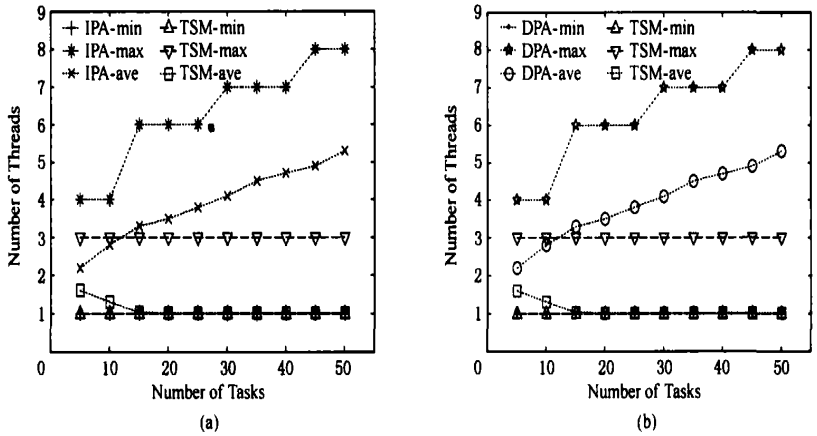

Fig. 2 Number of threads used by algorithms, with $maxPeriod = 100$. (a) Comparison between TSM and IPA and (b) Comparison between TSM and DPA.

图2 $maxPeriod = 100$ 时算法所用线程个数 (a) TSM与IPA算法的比较; (b) TSM与DPA算法的比较

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, IPA和DPA算法会使用比TSM算法多的系统优先级.

(2) $maxPeriod = 1000$ 时的结果

图3(a)和图3(b)分别给出 $maxPeriod = 1000$ 时TSM算法与IPA和DPA算法的性能比较. 可以看出, 结果与 $maxPeriod = 100$ 时相似. 所以在大部分情况下, IPA算法和DPA算法会使用较多的系统优先级.

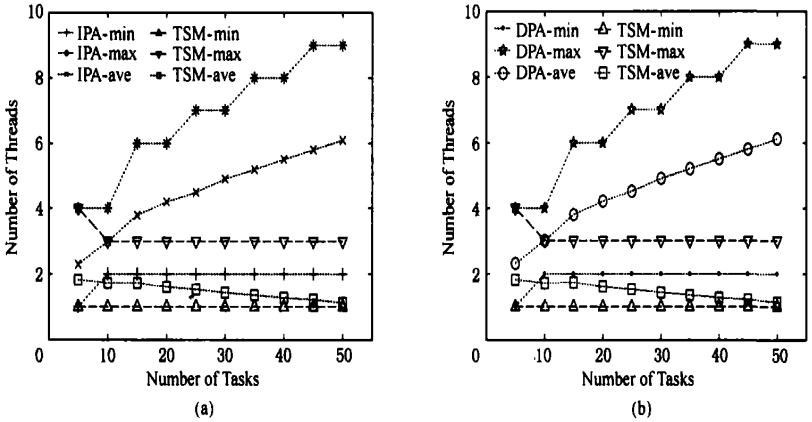

Fig. 3 Number of threads used by algorithms, with $maxPeriod = 1000$. (a) Comparison between TSM and IPA and (b) Comparison between TSM and DPA.

图3 $maxPeriod = 1000$ 时算法所用线程个数 (a) TSM与IPA算法的比较; (b) TSM与DPA算法的比较

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

TSM 算法没有引起新的阻塞时间,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提高任务集合的可调度性. 所以,与 IPA 和 DPA 算法相比,TSM 算法能在提高任务集合可调度性的同时,使用较少的系统优先级. 对于现有的优先级映射算法来说,这是两个相互矛盾不能同时实现的性能指标. 但 TSM 算法需要事件驱动线程框架的配合,从而增加了实现应用程序的复杂度.

6 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. 现有映射算法进行优先级映射时,在保证系统可调度的前提下,往往需要较多的优先级. 本文通过分析抢占阈值调度模型中,非抢占组生成方法的特点,提出了阈值段间映射法和事件驱动线程框架. 在不影响任务集合可调度性的前提下,将所用系统优先级减少到系统线程个数. 而且只需在优先级映射前判定任务集合的可调度性. 与现有映射算法的仿真比较表明:大多数情况下,TSM 算法使用的优先级个数少于 IPA 和 DPA 算法. 本文提出的算法和线程框架对实时嵌入式系统的设计和实现有一定的参考价值.

参 考 文 献

1 J. P. Lehoczky, L. Sha, Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In: Proc. Real-Time Systems Symp. Los Alamitos, CA: IEEE Computer Society Press, 1989. 166~171

2 C. L. Liu, J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. Journal of the ACM, 1973, 20(1): 46~61

3 J. P. Lehoczky, L. Sha. Performance of real-time bus scheduling algorithms. ACM SIGMETRICS Performance Evaluation Review, 1986, 14(1): 44~53

4 J. Orozco, R. Cayssials, J. Santos, *et al*. On the minimum number of priority levels required for the rate monotonic scheduling of real-time systems. The 10th EUROMICRO Workshop on Real Time Systems, Berlin, 1998

5 D. I. Katcher, S. S. Sathaye, J. K. Strosnider. Fixed priority scheduling with limited priority levels. IEEE Trans. Computers, 1995, 44(9): 1140~1144

6 L. C. Dpippo, V. F. Wolfe, L. Esibov, *et al*. Scheduling and priority mapping for static real-time middleware. Real-Time Systems, 2001, 20(2): 155~182

7 X. L. Bin, Y. H. Yang, J. Shiyao. Optimal fixed priority assignment with limited priority levels. In: Lecture Notes in Computer Science 2834. Berlin: Springer, 2003. 194~203

8 M. Saksena, Y. Wang. Scalable real-time system design using preemption thresholds. The 21st IEEE Real-Time Systems Symposium, Orlando, FL, 2000

9 Y. Wang, M. Saksena. Scheduling fixed-priority tasks with preemption threshold. The 16th Int'l Conf. Real-Time Computing Systems and Applications (RTCSA'99), Hong Kong, 1999

10 C. M. Bailey, A. Burns, A. J. Wellings, *et al*. A performance analysis of a hard real-time system. Department of Computer Science, University of York, Tech. Rep.: YCS 224, 1994

Wang Baojin, born in 1974. Ph. D. His main research interests include real-time system, scheduling algorithms, and software defined radio.

王保进,1974 年生,博士,主要研究方向为实时系统、调度算法和软件无线电技术.

Li Mingshu, born in 1966. Ph. D., professor and Ph. D. supervisor. His main research interests include software requirement engineering, software process, and real-time systems.

李明树,1966 年生,博士,研究员,博士生导师,主要研究方向为需求工程、软件过程和实时系统.

Research Background

When the priority levels that a task set requires are more than the static priority scheduling system can support, more than one task must be assigned the same priority. The priority mapping algorithms that have been used can augment the worst-case response time of task with higher priority, and may reduce the system schedulability. If keeping system schedulable, these algorithms may require more priority levels than the system can support. With preemption threshold scheduling model, the schedulability is improved as compared with both preemption and non-preemption scheduling models, and all tasks can be handled with the minimum number of event handling threads. But, the number of system priority levels that a thread needs does not be reduced. This paper presents a priority mapping algorithm called TSM (threshold segment mapping) and a thread implementation architecture with event-driven. It is shown that the TSM algorithm can maintain the strict order of tasks' priorities, and does not improve task's worst-case response time. The number of system priority levels used by algorithm and architecture equals to that of threads. Simulations show, when maintaining system schedulability, the TSM algorithm uses less priority levels than the existing priority mapping algorithms. Our work is supported by the National High Technology Research and Development Program of China (No. 2004AA1Z2050).

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net