ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.7, July 2007, pp.1844−1854
DOI: 10.1360/jos181844

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

一种静态最少优先级分配算法*

邢建生[1,2,3+], 王永吉[1,2], 刘军祥[1,2,3], 曾海涛[1,2,3], NASRO Min-Allah[1,2,3]

[1](中国科学院 软件研究所 互联网软件技术实验室,北京 100080)

[2](中国科学院 软件研究所 计算机科学国家重点实验室,北京 100080)

[3](中国科学院 研究生院,北京 100049)

A Static Priority Assignment Algorithm with the Least Number of Priority Levels

XING Jian-Sheng[1,2,3+], WANG Yong-Ji[1,2], LIU Jun-Xiang[1,2,3], ZENG Hai-Tao[1,2,3], NASRO Min-Allah[1,2,3]

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

[2](State Key Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

[3](Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

+ Corresponding author: Phn: +86-10-62661662 ext 1005, E-mail: jiansheng@itechs.iscas.ac.cn, http://itechs.iscas.ac.cn

Xing JS, Wang YJ, Liu JX, Zeng HT, Nasro MA. A static priority assignment algorithm with the least number of priority levels. *Journal of Software*, 2007,18(7):1844−1854. http://www.jos.org.cn/1000-9825/18/1844.htm

Abstract: With the increased penetration of real time systems into rapidly evolving systems, especially, in on-chip systems such as PDA (personal digital assistant) and PSP (play station portable) etc., the performance/price ratio is becoming a major concern for the system designers. At present, to maximize the performance/price ratio, these systems provide a limited number of priority levels to reduce price and the same priority level is assigned to multiple tasks. Such a trade off makes the most widely used static priority assignment algorithms such as RM (rate monotonic) and DM (deadline monotonic), impractical. As an alternative approach, static limited priority assignment assigns priority to tasks in such a way that the system feasibility is maintained by employing only a few priority levels. To date, static limited priority assignments can be classified into two categories, fixed-number priority and least-number priority assignment. This paper proposes a necessary and sufficient condition for analyzing task set feasibility under static limited priority assignment to make it suitable for a wide rage of applications. In addition, the superiority of low-to-high priority assignment strategy and the concept of saturated task group/assignment are highlighted along with several important properties for transformation among the limited priority assignments. A formal proof is drawn in favor of the least-number priority assignment when tasks are static

* Supported by the National Natural Science Foundation of China under Grant Nos.60373053, 60673121 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2003AA1Z2220 (国家高技术研究发展计划(863)); the State Education Ministry's Scientific Research Foundation for the Returned Overseas Chinese Scholars (国家教育部留学回国人员科研启动基金); the Research Collaboration between the Chinese Academy of Sciences and the Royal Society of the United Kingdom under Grant Nos.20030389, 20032006 (中国科学院与英国皇家学会联合资助项目); the Plan of Hundreds Scientists in the Chinese Academy of Sciences (中国科学院百人计划)

Received 2006-06-10; Accepted 2007-01-23

priority schedulable. Finally, a least-number priority assignment algorithm with low time complexity is presented and its efficiency is verified by the experimental results.

Key words: real-time system; static priority scheduling; limited priority level; priority assignment; schedulability test; performance/price ratio

摘 要: 随着实时系统越来越多地应用于各种快速更新系统,尤其是各种片上系统,如 PDA(personal digital assistant),PSP(play station portable)等,性价比已成为系统设计者的主要关注点.实际应用中,实时系统通常仅支持较少的优先级,常出现系统优先级数小于任务数的情况(称为有限优先级),此时,需将多个任务分配到同一系统优先级,RM(rate monotonic),DM(deadline monotonic)等静态优先级分配算法不再适用.为此,静态有限优先级分配是研究在任务集合静态优先级可调度的情况下,可否以及如何用较少或最少的系统优先级保持任务集合可调度.已有静态有限优先级分配可分为两类:固定数目优先级分配和最少优先级分配.给出了任意截止期模型下任务静态有限优先级可调度的充要条件以及不同静态有限优先级分配间转换时的几个重要性质,指出了系统优先级从低到高分配策略的优越性,定义了饱和任务组与饱和分配的概念,证明了在任务集合静态优先级可调度的情况下,最少优先级分配比固定数目优先级分配更具一般性.最后提出一种最少优先级分配算法 LNPA(least-number priority assignment).与现有算法相比,LNPA 适用范围更广,且复杂度较低.

关键词: 实时系统;静态优先级调度;有限优先级;优先级分配;可调度判定;性价比
中图法分类号: TP316 文献标识码: A

静态优先级调度具有算法实现简单、调度额外开销小、系统过载时可预测性好等优点,在实时系统中获得了广泛的应用.静态优先级调度的两个基本问题是优先级分配和可调度判定.Liu 和 Lanland 研究了隐含截止期(截止期等于周期)模型的静态优先级调度,提出了一种最优静态优先级分配算法 RM(rate monotonic)[1],大量文献研究了基于 RM 算法的可调度判定,文献[2]对这些研究进行了总结,文献[3]对相应算法进行了测试和比较.由于 RM 算法的最优性仅限于隐含截止期模型,为扩展静态优先级调度的适用范围,相继有人研究了受限截止期(截止期小于等于周期)模型和任意截止期(截止期与周期无关)模型的静态优先级调度,给出了受限截止期模型的最优静态优先级分配算法 DM(deadline monotonic)及其可调度判定条件[4],以及任意截止期模型一种特例情况的最优静态优先级分配算法 MRM(modified rate monotonic)及其可调度判定条件[5].

上述静态优先级分配算法均基于如下假设:系统支持足够多的优先级,每个任务均可获得唯一的系统优先级.对于专用实时系统而言,上述假设是合理的,因为对于专用实时系统,最重要的是实时性和可靠性,其次才考虑性价比.

近年来,随着市场需求的不断增加,实时系统已广泛应用于各种快速更新系统,特别是片上系统,如 PDA(personal digital assistant),PSP(play station portable)等,而性价比是衡量这些系统优劣的重要因素.为降低成本,在实际应用中,实时系统往往仅支持较少的优先级,如实时总线标准 MSI STD BUS 支持 2 个优先级[6],实时 POSIX 支持 32 个优先级[7],嵌入式操作系统 OS68 支持 32 个优先级[8],实时操作系统 LinuxRT 支持 256 个优先级[9].常出现系统优先级数小于任务数的情况,这种情况称为有限优先级,此时,须将多个任务分配到同一系统优先级,因此上述静态优先级分配算法不再适用,需研究有限优先级下的静态优先级分配(简称静态有限优先级分配).

记系统支持的优先级数为 m,任务集合包含的任务数为 n.静态有限优先级分配主要研究如下问题:在任务集合静态优先级可调度的情况下,可否以及如何用较少或最少的系统优先级保持任务集合可调度,从而最大化系统的性价比.相应地,静态有限优先级分配可分为两类:固定数目优先级分配和最少优先级分配.固定数目优先级分配研究可否以及如何用较少的 $m(m\leq n)$个系统优先级保持任务集合可调度;最少优先级分配研究可否以及如何用最少($\leq m$)的系统优先级保持任务集合可调度.文献[10]提出一种常量比例栅格法,本文简记为 CRG(const ratio grid)算法,该算法属于固定数目优先级分配,实现简单,但仅适用于隐含截止期模型,且分配后任

务集合的可调度性下降很多.文献[11]针对受限截止期模型,给出了有限优先级情况下任务集合可调度的充分必要条件,并指出最优优先级分配的时间复杂度呈指数形式,但未给出相应的算法.文献[12]给出了 RM 算法所需最少优先级的条件,但该条件仅适用于隐含截止期模型,且其也未给出相应的算法.文献[13]提出一种最少优先级分配算法,该算法复杂度低,但仅适用于隐含截止期模型,本文称其为 RM-Least.文献[14]提出一种 AGP(assignment of priority group)算法,它也属于最少优先级分配,可适用于隐含截止期、受限截止期以及文献[5]中任意截止期模型的特例,但其一般情况下复杂度太高.文献[15]提出了利用抢占阈值调度模型[16]生成的非抢占组的特性进行优先级分配的 TSM(threshold segment mapping)算法,TSM 算法不引入新的阻塞时间,可提高任务集合的可调度性,但该算法需要一个应用层的事件驱动线程框架来配合,其实际应用受到很大的限制.

针对现有研究的不足,讨论了两类静态有限优先级分配之间的关系(本文不考虑抢占阈值调度),并指出:在任务集合静态优先级可调度的情况下,最少优先级分配比固定数目优先级分配更具一般性.提出一种最少优先级分配算法 LNPA(least-number priority assignment).与已有算法相比,该算法适用范围更广,可适用于任意截止期模型,且复杂度较低.

1 任务模型、相关概念和研究

1.1 任务模型

本文基于任意截止期模型,讨论静态有限优先级分配,具体约定如下:

$\tau=\{\tau_1,\ldots,\tau_n\}$是含有 n 个周期任务的集合.

任务以 $\tau_i=\{T_i,C_i,D_i\}$表示,其中,T_i,C_i,D_i 分别表示 τ_i 的周期、执行时间和截止期(称作相对截止期).D_i 与 T_i 无关,可取任意值.

τ_i 的一次执行称作一个实例,τ_i 的第 k 个实例为 $\tau_i(k)$,其释放时刻为 $S_i(k)$.显然,$S_i(k)=kT_i$(k 从 0 开始),$\tau_i(k)$的截止期 $d_i(k)=kT_i+D_i$(称作绝对截止期).

本文假定:

(a) 各任务相互独立(不存在因共享资源而导致的阻塞);

(b) 任务不会自动挂起;

(c) 任务切换时间很小,可以忽略(假定为 0);

(d) 当同时存在一个任务的多个实例时,按其释放次序依次执行,即后来的实例不会影响之前的实例.

为便于区分,称由 RM,DM,MRM 等最优静态优先级分配算法分配的优先级为自然优先级,称静态有限优先级分配中,每个任务实际分配的优先级 $P_i\in[1,m]$为系统优先级.在实际系统中,常用队列管理系统优先级相同的任务,具有相同系统优先级的所有任务构成一个任务组(队列),在任务组内采取时间片轮转或先来先服务的方式调度.本文仅讨论了时间片轮转的方式,但相关结论也可推广到先来先服务方式.本文以 g_z 表示系统优先级为 Z 的任务组.优先级的值以正整数表示,数越大,则其表示的优先级越低,1 表示最高的(系统或自然)优先级.

1.2 相关概念

定义 **1**. 若一个任务组中所有任务都可调度,则称此任务组可调度;若一个任务组中至少有一个任务不可调度,则称此任务组不可调度.

不失一般性,定义空任务组也可调度.

定义 **2**. 若一个任务集合被分配在 l 个非空任务组中,且这 l 个任务组都可调度,则称此任务集合 l 个系统优先级可调度,当 $l\le m$(m 的含义见引言)时,称此任务集合静态有限优先级可调度.

为与传统的静态优先级可调度的概念有所区分,上面给出了静态有限优先级可调度的概念.显然,任务集合静态优先级可调度等价于其 n 个系统优先级可调度.

定义 **3**. 称先分配最低系统优先级然后依次分配较高系统优先级的静态有限优先级分配策略为系统优先级从低到高的策略;相应地,称先分配最高系统优先级然后依次分配较低系统优先级的策略为系统优先级从高

到低的策略.

1.3 相关研究

CRG 算法原理如下:根据任务集合的周期范围设定一个栅格$\{L_0,L_1,\ldots,L_m\}$,其中,L_0 表示可能的最小周期;L_m 表示可能的最大周期;常量比例 $r=\max(L_i/L_{i-1}),1\le i\le m$,表示相邻两个栅格的最大比例.若某任务周期 T 满足 $L_{i-1}<T\le L_i$,则给其分配系统优先级 i.该算法实现简单,但仅适用于隐含截止期模型.而且 CRG 算法未考虑任务的可调度性,不能确保正确回答可否以及如何以较少的系统优先级保持任务集合可调度的问题,参考价值不大.

RM-Least 算法遵循系统优先级从高到低的策略,它以 RM 算法得到的自然优先级分配为基础,按照自然优先级从高到低的顺序,依次考察任务的最大响应时间,将最大响应时间小于等于当前最高自然优先级任务截止期的任务都分配到最高系统优先级,然后考察次高系统优先级.依此类推.该算法复杂度低,为 $o(n)$,但其仅适用于隐含截止期模型.

AGP 算法也遵循系统优先级从高到低的策略.它首先由 RM,DM,MRM 等算法得到自然优先级分配,将自然优先级最高的任务分配到系统优先级最高的任务组,然后根据任务可调度的充要条件,判定次高自然优先级任务能否分配到系统优先级最高的任务组.若能,则将次高自然优先级任务分配到系统优先级最高的任务组;否则,将其分配到系统优先级次高的任务组.依此类推.对于基本任务集合,AGP 算法复杂度为 $o(n^2)$;但一般情况下,任务集合都是扩展任务集合,此时,AGP 算法复杂度很高,为 $o(n^2 C_n^m)$[14].

2 理论基础

2.1 任务静态有限优先级可调度充要条件

目前,对任务静态有限优先级可调度判定的研究还不够充分,文献[11]给出了受限截止期模型的可调度充要条件,文献[14]给出了截止期大于周期情况下的可调度充要条件.本节基于经典的 *level–i* 忙周期分析方法[17−19],对任意截止期模型进行可调度分析,给出相应的可调度充要条件.

可调度分析基于对任务最大响应时间的计算,如果一个任务的最大响应时间小于等于其截止期,则该任务可调度;如果任务集合中所有任务都可调度,则该任务集合可调度.对于任意截止期模型,τ_i 的最大响应时间不一定发生在其临界时刻后第 1 个实例上,需要考察其 *level–i* 忙周期内的所有实例.所谓 *level–i* 忙周期即一个连续的时间段,在此时间段只有系统优先级高于等于 τ_i 的任务才可以运行,*level–i* 忙周期起始于临界时刻,临界时刻发生在所有系统优先级高于等于 τ_i 的任务同时释放(通常假定为时刻 0)时.

考虑 g_z 中的任务 τ_i,以 $W_i(k)$表示 $\tau_i(k)$的完成时间,从临界时刻到 $W_i(k)$的时间段内,须执行 τ_i 以及所有系统优先级高于等于 τ_i 的任务.系统优先级高于 τ_i 的任务执行时间为

$$\sum_{\tau_j\in g_p,p<z}\left\lceil W_i(k)/T_j\right\rceil C_j\ .$$

系统优先级等于 τ_i 的任务执行时间为

$$\sum_{\tau_j\in g_p,p=z,\tau_j\ne\tau_i}\left\lceil W_i(k)/T_j\right\rceil C_j\ .$$

在 g_z 中,采取时间片轮转的方式调度,每个任务都可能最后执行.不失一般性,应考虑最坏情况,即 $\tau_i(k)$最后执行.

τ_i 执行时间为$(k+1)C_i$.因此,$W_i(k)$可由下式递归求得,

$$W_i(k)=\sum_{\tau_j\in g_p,p\le z,\tau_j\ne\tau_i}\left\lceil W_i(k)/T_j\right\rceil C_j+(k+1)C_i \tag{1}$$

τ_i 的忙周期 W_i 可由下式表示,

$$W_i=\min_{k\in\{1,2,\ldots\}}W_i(k)::W_i(k)\le(k+1)T_i \tag{2}$$

考察忙周期 W_i 内 τ_i 所有实例的响应时间,其最大值即为 τ_i 最大响应时间 R_i,可知 τ_i 可调度的充要条件为

$$R_i = \max_{k=0,\ldots,l}(W_i(k) - kT_i) \le D_i \quad (3)$$

其中,l 是满足式(2)的 k 的最小值.

2.2 不同静态有限优先级分配间转换时的几个重要性质

定理 **1**. 若 g_z 可调度,则将任何系统优先级高于 g_z 的任务加入 g_z,g_z 中原有任务仍可调度.

证明:任取一个系统优先级高于 g_z 的任务 τ_x,任取一个 g_z 的任务 τ_i.据上节充要条件可知,τ_i 可调度当且仅当其满足式(3),由式(1)可知,将 τ_x 加入 g_z 前的 $W_i(k)$与将 τ_x 加入 g_z 后的 $W_i(k)$相同,因此,忙周期 W_i 不变.忙周期 W_i 中 τ_i 任一实例的响应时间也不变,由式(3)可知,τ_i 的最大响应时间 R_i 也不变,τ_i 仍可调度.因 τ_i 是任取的,可知 g_z 原有任务仍可调度,定理得证. □

定理 **2**. 若 g_z 可调度,则将任何系统优先级高于 g_z 的任务移到系统优先级低于 g_z 的任务组,g_z 仍可调度.

证明:与定理 1 类似. □

定理 **3**. 若 g_z 可调度,则将任何 g_z 的任务移到系统优先级低于 g_z 的任务组,g_z 中其他任务仍可调度.

证明:与定理 1 类似. □

2.3 系统优先级从低到高的分配策略

RM-Least 和 AGP 算法均采用了系统优先级从高到低的分配策略,在从未分配的任务(其系统优先级低于或等于已分配任务)中选择一个任务,判定是否可为其分配当前系统优先级时,需重新考察已分配到当前系统优先级任务的可调度性.若采用系统优先级从低到高的分配策略,则从未分配的任务(其系统优先级高于或等于已分配任务)中选择一个任务.判定是否可为其分配当前系统优先级时,据定理 1 可知,无须考察已分配到当前系统优先级任务的可调度性.因此,与系统优先级从高到低的分配策略相比,系统优先级从低到高的分配策略可以极大地减少任务可调度判定的次数.

3 最少优先级分配

静态有限优先级分配的主要目的是在任务集合静态优先级可调度的情况下,以较小或最小的代价(较少或最少的优先级)保持任务集合可调度,从而最大化系统的性价比.因此,在后续讨论中均假定任务集合静态优先级可调度.

3.1 两类静态有限优先级分配之间的关系

定理 **4**. 若任务集合 $l(l<n)$个系统优先级可调度,则其也一定是 $l+1$ 个系统优先级可调度.

证明:在任务集合静态优先级可调度的情况下,必存在可行的静态优先级分配,设其为$\{\tau_1,\ldots,\tau_n\}$(其中的下标也表示各任务的优先级).若该任务集合 l 个系统优先级可调度,记其为 $g_1,g_2,\ldots,g_l$.在此分配中,将 τ_n 移到新增的任务组 g_n,得到新任务组 $g_1', g_2',\ldots, g_l', g_n$,由定理 2、定理 3 可知,$g_1', g_2',\ldots, g_l'$ 仍可调度.g_n 只有一个任务 τ_n,而判定 τ_n 可调度的条件与非有限优先级情况下相同,故 τ_n 可调度,g_n 也可调度.若 $g_1', g_2',\ldots, g_l'$ 中不含空任务组,则任务集合 $l+1$ 个系统优先级可调度;若 $g_1', g_2',\ldots, g_l'$ 中含空任务组,则将其删除,得到新任务组 $g_1'', g_2'',\ldots, g_{n-1}''$,然后将 τ_{n-1} 移到新增的任务组 g_{n-1},重复上述推理.依此类推,可知任务集合一定是 $l+1$ 个系统优先级可调度.定理得证. □

定理 **5**. 若任务集合 $l(l<n)$个系统优先级可调度,则其也一定是 $l+1,\ldots,n$ 个系统优先级可调度.

证明:在定理 4 的基础上递推或由归纳法,容易证明本定理成立,此处从略. □

由定理 5 可知如下定理成立:

定理 **6**. 若任务集合最少优先级 $l(l<n)$可调度,则其也一定是固定优先级 $m(l\le m)$可调度.

由定理 6 可知,相比固定数目优先级分配而言,最少优先级分配更具一般性.

3.2 饱和任务组与饱和分配

为有效应用系统优先级从低到高的分配策略,定义如下饱和任务组与饱和分配的概念.

定义 **4**. 已知 g_z 可调度,若系统优先级高于 g_z 的任务中存在一个或多个任务,将其加入 g_z 后,新任务组仍可调度,则称 g_z 为不饱和任务组;否则,称 g_z 为饱和任务组.

由上述定义可知,饱和任务组一定可调度;反之则不然.

定义 **5**. 称所有任务组都是饱和任务组的静态有限优先级分配为饱和分配.

定理 **7**. 已知 g_z 可调度,如果存在多个任务,将其加入 g_z 后,新任务组可调度,则一定存在一个任务,将其加入 g_z 后,新任务组可调度.

证明:如果存在多个任务 $\tau_a,\tau_b,\ldots$,将其加入任务组 g_z 后,得到的新任务组可调度.显然,$\tau_a,\tau_b,\ldots$都可调度.在这多个任务中,任取一个任务 τ_a,可知 τ_a 可调度,其最大响应时间小于等于其截止期.由式(1)、式(2)可知,此时,τ_a 的忙周期与将 τ_a 单独加入 g_z 时的忙周期相同,且其每个实例的响应时间与将 τ_a 单独加入 g_z 时的也相同,因此,由式(3)可知,将 τ_a 单独加入 g_z,τ_a 的最大响应时间也小于等于其截止期,τ_a 也可调度.而由定理 1 可知,将 τ_a 单独加入 g_z,g_z 中原有任务仍可调度.综上可知,新任务组也可调度.因 τ_a 是任取的,可知定理成立. □

由定理 7 可知其逆否命题成立:已知 g_z 可调度,如果不存在一个任务,将其加入 g_z 后,新任务组可调度,则也不存在多个任务,将其加入 g_z 后,新任务组可调度.因此,在判定任务组 g_z 是否饱和时,只须判定系统优先级高于 g_z 的任务中是否存在一个任务,将其加入 g_z 后,新任务组仍可调度,若不存在,则 g_z 是饱和任务组;否则,g_z 是不饱和任务组.

定理 **8**. 饱和分配是最少优先级分配.

证明:采用反证法.

假设存在可行的、使用更少系统优先级的分配,则与其相比,饱和分配(记为 $g_l,g_{l+1},\ldots,g_m$,其中,$l\geq 1$)中至少有一个任务组中的任务要被分布到其他任务组.此任务组不可能是 g_l(因系统优先级比 g_l 低的任务组都是饱和的).设此任务组为 $g_i(i>l)$.显然,g_i 中的任务不能添加到 $g_{i+1},\ldots,g_m$ 中,只能将 g_i 中的任务分布到 $g_l,\ldots,g_{i-1}$ 中,由假设可知,转换后,新任务组 $g'_1,\ldots,g'_{i-1}$ 仍可调度.在此基础上,将所有属于 g_i 的任务都移到 g'_{i-1} 中,得到新任务组 $g''_1,\ldots,g''_{i-1}$,据定理 2、定理 3 可知,$g''_1,\ldots,g''_{i-2}$ 仍可调度.而 $g''_{i-1}=g'_{i-1}\cup g_i$,由定理 1 可知,原先属于 g'_{i-1} 的任务仍可调度.而 g_i 是 $g''_1,\ldots,g''_{i-1}$ 组成的任务子集中,最低系统优先级可调度的,由定理 1 可知,原先属于 g_i 的任务也可调度.综上可知,g''_{i-1} 可调度,所以,$g''_{i-1}\subseteq g_i$(否则与 g_i 是饱和的事实矛盾).已知 $g''_{i-1}=g'_{i-1}\cup g_i$,而 $g'_{i-1}\supseteq g_{i-1}$,因为 g_{i-1} 与 g_i 不相交,所以 $g''_{i-1}\supset g_i$,与前一结论 $g''_{i-1}\subseteq g_i$ 相矛盾.所以假设不成立,定理得证. □

3.3 最少优先级分配算法

本节给出一种最少优先级分配算法 LNPA(least-number priority assignment).它采用系统优先级从低到高的分配策略,并且产生的任务组都是饱和的.下面为算法描述.

INPUT: A given task set τ and the number of provided priority levels m;

OUTPUT: A limited priority assignment Φ or an assertion that the task set is not limited priority feasible.

```
1.   τ*=τ                  /* duplicate of the task set */
2.   g=m                   /* g is the current system priority level */
3.   n=length(τ*)
4.   While (n>1)
5.       For τj in τ*
6.           If (schedulable(τj,g)
7.               Φ(τj)=g
8.               τ*=τ*−τj
9.               n=n−1
10.          End If
11.      End For
12.      If (n>1)
```

```
13.            g=g−1
14.          If (g<1)
15.                Exit
16.          End If
17.       End If
18.  End
```

初始时,置各系统优先级任务组为空,当前系统优先级为 m.在任务集合中,逐个寻找当前系统优先级可调度的任务,将其添加到当前系统优先级任务组,并将其从任务集合中删除,直到不存在当前系统优先级可调度的任务时,将 $m-1$ 置为当前系统优先级,重复上述工作.依此类推.

3.4 算法性质

定理 **9**. 已知 g_m 是饱和的,如果任务集合存在不包含于该任务组的任务,则一定存在一个任务,该任务系统优先级 $m-1$ 可调度.

证明:在任务集合静态优先级可调度的情况下,必存在可行的静态优先级分配,设其为 $g'_1,\ldots,g'_n$ (其中,各任务组有且仅有一个任务).在此分配中,将属于 g_m 的任务移到最低系统优先级,得到新任务组 $g''_1,\ldots,g''_n$,显然,$g''_n=g_m\cup g'_n$.据定理 1 可知,属于 g_m 的任务仍可调度.同理,属于 g'_n 的任务仍可调度.因此,g''_n 仍可调度.可知 $g''_n\subseteq g_m$(否则与 g_m 是饱和的事实相矛盾),而由 $g''_n=g_m\cup g'_n$ 可知,$g''_n\supseteq g_m$.综上可知,$g''_n=g_m$.据定理 2、定理 3 可知,$g''_1,\ldots,g''_{n-1}$ 仍可调度.这些任务组或者为空,或者有且仅有一个任务,当任务集合存在不包含于 g_m 的任务,即 $g''_1,\ldots,g''_{n-1}$ 中有非空任务组时,其中系统优先级最低的任务组中的任务即为系统优先级 $m-1$ 可调度的任务.定理得证. □

定理 **10**. 已知 $g_i,\ldots,g_m$ 都是饱和的,如果任务集合中存在不包含于上述任务组的任务,则一定存在一个任务,该任务系统优先级 $i-1$ 可调度.

证明:在定理 9 的基础上容易证明本定理成立,本文从略.

由算法描述可知,LNPA 得到的分配都是饱和分配.由定理 9、定理 10 可知,LNPA 的执行只能出现两种情况:所有任务分配完毕,或者在当前系统优先级小于 1 时停止分配.下面分别对这两种情况进行讨论.

第 1 种情况:

所有任务分配完毕.设最后分配的系统优先级为 i,则其所需系统优先级数为 $m-i+1$,由定理 8 可知,$m-i+1$ 即任务集合所需最少优先级数,相应的分配即为此任务集合的最少优先级分配.

第 2 种情况:

在当前系统优先级小于 1 时停止分配.这说明任务集合所需最少优先级数大于系统支持的优先级数 m,因此,任务集合静态最少优先级不可调度,同时也静态有限优先级不可调度.

因此,LNPA 算法具有如下性质:

性质 **1**. 若 LNPA 得到一个完整的分配,则相应的任务集合静态最少优先级可调度,且该分配即为其最少优先级分配.

性质 **2**. 若 LNPA 不能得到一个完整的分配,则该任务集合静态最少优先级不可调度.

综上可知,LNPA 算法同时回答了可否以及如何用最少的系统优先级保持任务集合可调度的问题.

3.5 算法扩展

由上文可知,最少优先级分配更具一般性.相应地,本文算法也可进行扩展,以适用于固定数目优先级分配.具体如下:若进行固定数目优先级分配,在找不到可添加到当前系统优先级任务组的任务,须将当前系统优先级置为较高系统优先级时,先比较未分配的系统优先级数与未分配的任务数,若未分配的任务数较大,则算法不变;否则,在继续分配时,只需寻找一个当前系统优先级可调度的任务,无须满足任务组饱和的条件(仿照上节的推理过程,容易证明,该固定数目优先级分配算法也可以同时解决可否以及如何用较少的系统优先级保持任务

集合可调度的问题).

4 算法比较与分析

因 CRG 算法参考价值不大,本文仅将 LNPA 算法与现有的两种最少优先级分配算法进行比较.

4.1 算法执行实例

为直观地理解算法,本节给出一个例子,详细说明了现有算法和本文算法进行静态最少优先级分配的过程.

例 1:表 1 给出一个含有 10 个任务的集合,为便于各种算法的比较,任务集合属于隐含截止期模型.表中 Natural priority 是由 RM 算法分配的自然优先级,WCRT(worst case response time)表示任务的最大响应时间.在后续各表中,Step 表示算法的执行步骤,System priority 表示为任务分配的系统优先级.

Table 1 Arguments of task set

表 1 任务集合参数

Natural priority	1	2	3	4	5	6	7	8	9	10
C_i	1	2	1	1	1	1	1	1	1	1
T_i	5	10	10	10	15	18	20	20	20	20
WCRT	1	3	4	5	7	8	9	10	18	20

表 2 列出了 RM-Least 算法的执行过程.Current task 是指当前任务,Compare 表示比较过程.因 τ_1 的周期≥τ_4 的 WCRT,所以为 τ_1~τ_4 分配系统优先级 1.同理,因 τ_5 的周期≥τ_8 的 WCRT,为 τ_5~τ_8 分配系统优先级 2,依此类推.

Table 2 Execution process of RM-Least

表 2 RM-Least 执行过程

Step	Current task	Compare	System priority
1	1	T_1>WCRT (τ_1)	1
2	2	T_1>WCRT (τ_2)	1
3	3	T_1>WCRT (τ_3)	1
4	4	T_1≥WCRT (τ_4)	1
5	5	T_5<WCRT (τ_5)	
6	5	T_5>WCRT (τ_5)	2
7	6	T_5>WCRT (τ_6)	2
8	7	T_5>WCRT (τ_7)	2
9	8	T_5>WCRT (τ_8)	2
10	9	T_5<WCRT (τ_9)	
11	9	T_9>WCRT (τ_9)	3
12	10	T_9≥WCRT (τ_{10})	3

表 3 列出了 AGP 算法的执行过程.表中 Schedulable task 是指根据上文充要条件,可加入当前任务组的任务(以其自然优先级表示).Current task group 是指当前任务组中已有任务(以其自然优先级表示).Max number of schedulability test 是指为加入 Schedulable task 须进行的可调度判定的次数.

Table 3 Execution process of AGP

表 3 AGP 执行过程

Step	System priority	Schedulable task	Current task group	Max number of schedulability test
1	1	1		1
2	1	2	1	2
3	1	3	1,2	3
4	1	4	1,2,3	4
5	1		1,2,3,4	5
6	2	5		1
7	2	6	5	2
8	2	7	5,6	3
9	2	8	5,6,7	4
10	2	9	5,6,7,8	5
11	2		5,6,7,8,9	6
12	3	10		1
13	3		10	0

表 4 列出了 LNPA 算法的执行过程.其各参数意义与表 3 相同.

Table 4 Execution process of LNPA algorithm

表 4 LNPA 执行过程

Step	System priority	Schedulable task	Current task group	Max number of schedulability test
1	3	7		10
2	3	8	7	9
3	3	10	7,8	8
4	3	9	7,8,10	7
5	3		7,8,9,10	6
6	2	3		6
7	2	2	3	5
8	2	5	2,3	4
9	2	6	2,3,5	3
10	2	4	2,3,5,6	2
11	2		2,3,4,5,6	1
12	1	1		1
13	1		1	0

4.2 算法性能比较与分析

LNPA 算法中,第 1 次分配系统优先级时,最多考察 n 个任务的可调度性,第 2 次分配系统优先级时,最多考察 n–1 个任务的可调度性,依此类推,最多进行 $n+(n-1)+\ldots+1=(n^2+n)/2$ 次可调度判定,故算法复杂度为 $o(n^2)$.

RM-Least 复杂度低,为 $o(n)$,但其仅适用于隐含截止期模型.AGP 算法适用范围较广,可用于隐含截止期、受限截止期以及任意截止期模型的一种特例[5],但仍不能适用于一般的任意截止期模型,且一般情况下,算法复杂度太高,为 $o(n^2 C_n^m)$.而 LNPA 算法适用范围广,可适用于任意截止期模型,且算法复杂度较低,为 $o(n^2)$.

对于扩展任务集合,AGP 算法复杂度太高,无法通过实验比较.本文通过如下实验,比较 LNPA 与 AGP 算法在基本任务集合(实验中产生的任务集合可能是基本任务集合,也可能是扩展任务集合.为便于比较,都按基本任务集合处理,显然这不影响算法性能)下的时间性能.图 1 给出了实验结果.实验条件如下:以任务集合中任务数分组,共生成 45 组任务集合.任务数从 5 依次递增到 50.每组集合中各有 100 个任务集合.任务集合中的任务满足以下条件:

(1) 任务周期在[10,10000]上均匀分布,任务的截止期等于周期.

(2) 任务集合静态优先级可调度.

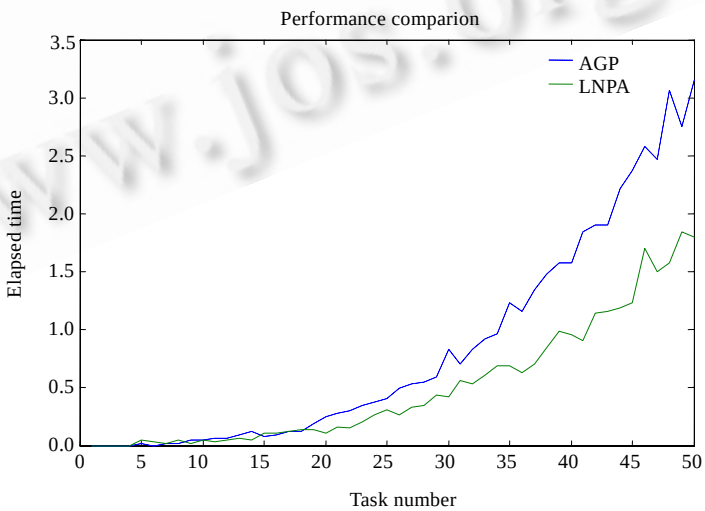

Fig.1 Execution time comparison of LNPA versus AGP

图 1 LNPA 与 AGP 算法运行时间比较

对于基本任务集合,AGP 算法复杂度为 $o(n^2)$,与 LNPA 算法复杂度相同.从图 1 可以看出,这两种算法时间性能是类似的,并且随着任务数的增加,LNPA 算法的实际性能要稍好于 AGP 算法.

5 结 论

随着实时系统在各领域,尤其是快速更新系统,如 PDA,PSP 等系统中的广泛应用,性价比已成为衡量系统优劣的重要因素,是系统设计者考虑的主要关注点之一.静态有限优先级分配主要研究当系统仅支持较少的优先级时,可否以及如何以较少或最少的优先级保持任务集合可调度(在任务集合静态优先级可调度的情况下),从而最大化系统性价比.针对现有研究的不足,本文将静态有限优先级分配分为两类,并指出:在任务集合静态优先级可调度的情况下,最少优先级分配比固定数目优先级分配更具一般性.本文给出了任意截止期模型的任务静态有限优先级可调度的充要条件以及不同静态有限优先级分配间转换时的几个重要性质,指出了系统优先级从低到高分配策略的优越性,定义了饱和任务组与饱和分配的概念.在此基础上,提出了一种最少优先级算法 LNPA.与 RM-Least 算法相比,LNPA 适用范围更广;与 AGP 算法相比,LNPA 适用范围更广,且一般情况下复杂度更低.本文算法对于在实时系统设计和实现时,提高系统性价比具有重要的指导意义.

References:

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

[2] Wang YJ, Chen QP. On schedulability test of rate monotonic and its extendible algorithms. Journal of Software, 2004,15(6): 799−814 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/799.htm

[3] Xing JS, Liu JX, Wang YJ. Schedulability test performance analysis of rate monotonic algorithm and its extended ones. Journal of Computer Research and Development, 2005,42(11):2025−2032 (in Chinese with English abstract).

[4] Audsley NC, Burns A, Richardson MF, Wellings AJ. Hard real-time scheduling: The deadline-monotonic approach. In: Proc. of the 8th IEEE Workshop on Real-Time Operating Systems. Oakland: IEEE Computer Society Press, 1991. 133−137.

[5] Shih WK, Liu JWS, Liu CL. Modified rate-monotonic algorithm for scheduling periodic jobs with deferred deadlines. IEEE Trans. on Software Engineering, 1993,19(12):1171−1179.

[6] MSI-C851 STD BUS 80C51 Microcontroller Card. Microcomputer systems. http://www.microcomputersystems.com

[7] Harbour MG. Real-Time POSIX: An overview. In: Proc. of the VVConex'93 Int'l Conf. 1993. http://www.ctr.unican.es/publications/mgh-19931.pdf

[8] Song XM. Real-Time operating system case study: OS68 for embedded systems. 1997. http://www.cs.helsinki.fi/u/kraatika/Courses/Old/RTS97/rtseminar_97_song.html

[9] The concise handbook of linux for embedded real-time systems version 1.1. Timesys Corporation, 2002. http://www.timesys.com

[10] Lehoczky JP, Sha L. Performance of real-time bus scheduling algorithms. In: Proc. of the '86 ACM SIGMETRICS Joint Int'l Conf. on Computer Performance Modeling, Measurement and Evaluation. New York: ACM Press, 1986. 44−53.

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

[12] 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

[13] Cayssials R, Orozco J, Santos J, Santos R. Rate monotonic scheduling of real-time control systems with the minimum number of priority levels. In: Proc. of the 11th Euromicro Conf. on Real Time Systems. Oakland: IEEE Computer Society Press, 1999. 54−59.

[14] Bin XL, Yang YH, Jin SY. An assignment algorithm of static priority for limited priority levels. Journal of Software, 2004,15(6): 815−822 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/815.htm

[15] Wang BJ, Li MS, Wang ZG. Uniprocessor static priority scheduling with limited priority levels. Journal of Software, 2006,17(3): 602−610 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/602.htm

[16] 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.

[17] Harbor MG, Klein MH, Lehoczky JP. Fixed priority scheduling of periodic tasks with varying execution priority. In: Proc. of the IEEE Real-Time Systems Symp. Oakland: IEEE Computer Society Press, 1991. 116−128.

[18] Lehoczky JP. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In: Proc. of the 11th Real-Time Systems Symp. Oakland: IEEE Computer Society Press, 1990. 201−209.

[19] Tindell KW, Burns A, Wellings A. An extendible approach for analyzing fixed priority hard real-time tasks. The Journal of Real-Time Systems, 1994,6(2):133−152.

附中文参考文献:

[2] 王永吉,陈秋萍.实时单调速率及其扩展算法的可调度性判定.软件学报,2004,15(6):799−814. http://www.jos.org.cn/1000-9825/15/799.htm

[3] 邢建生,刘军祥,王永吉.RM 及其扩展可调度性判定算法性能分析.计算机研究与发展,2005,42(11):2025−2032.

[14] 宾雪莲,杨玉海,金士尧.一种有限优先级的静态优先级分配算法.软件学报,2004,15(6):815−822. http://www.jos.org.cn/1000-9825/15/815.htm

[15] 王宝进,李明树,王志刚.优先级有限时的单处理器静态优先级调度.软件学报,2006,17(3):602−610. http://www.jos.org.cn/1000-9825/17/602.htm

邢建生(1972－),男,河南安阳人,博士生,主要研究领域为实时系统,嵌入式系统.

王永吉(1962－),男,博士,研究员,博士生导师,主要研究领域为实时系统,网络优化,智能软件工程,非线性优化理论,实时混合控制理论,实时嵌入式操作系统.

刘军祥(1975－),男,博士,主要研究领域为网络优化理论,实时调度算法.

曾海涛(1979－),男,博士生,主要研究领域为安全操作系统,实时调度算法.

Nasro Min-Allah(1975－),男,博士生,主要研究领域为实时系统,节能调度.