第34卷 第8期 北京工业大学学报 Vol.34 No.8
2008年 8月 JOURNAL OF BEIJING UNIVERSITY OF TECHNOLOGY Aug. 2008

基于有限优先级的动态调度分组算法

何福贵，侯义斌，李 辉
（北京工业大学 软件学院 嵌入式软件与系统研究所，北京 100022）

摘 要：实时调度是实时系统中的关键问题，实时动态调度是实时调度的主要方面．实时调度在理论分析时，都假设系统能识别任意多的优先级．当实时调度应用于实际的任务系统时，仅能使用有限的优先级数量．在实际的任务系统中进行动态调度分析时包含任务系统动态调度所需的最小优先级数量的判断方法和任务系统分组算法．在此基础上，给出了任务系统分组的算法及最优分组的判定条件，并详细说明了任务系统分组算法的步骤和过程．

关键词：动态调度；有限的优先级数量；任务分组算法；实时调度
中图分类号：TP 316.2 **文献标识码**：A **文章编号**：0254－0037(2008)08－0873－05

实时系统是一种计算机系统，计算的正确性不仅依赖于计算结果，而且依赖于产生结果的时间．实时系统的应用已变得越来越广泛．实时调度是实时系统中的核心，分为静态调度和动态调度．实时调度算法在过去几十年进行了广泛的研究，静态调度算法主要有单调速率(rate-monotonic，简称为RM)算法[1]和单调截止期(deadline monotonic，简称为DM)算法[2]，动态调度算法主要有截止期最早最优先(earliest deadline firs，简称为EDF)算法[1]、空闲时间最短最优先[3](least slack first，简称为LFS)算法和价值最高最优先[4](highest value first，简称为HVF)等．动态调度算法是一种在线调度算法，调度可行性判断在任务的执行过程中完成，需要较少的任务信息；静态调度算法是一种离线度算法，在运行中有较少的调度运行开销，但它的CPU利用率较低，且灵活性较差；RM算法是优化的静态调度算法，当系统中任务足够多时，CPU的利用率在0.69以下；动态调度算法是一种在线调度算法，它的调度开销较大，但它灵活，CPU的利用率较高；EDF是优化的动态调度算法，其CPU利用率接近1．

静态调度和动态调度适用于不同的场合，如果在调度前已知任务组中任务的属性值，那么就可以使用静态调度，如果任务组中任务的属性值需要在调度过程中计算，那么就必须使用动态调度．另外，如果任务组中任务较多且CPU利用率大于0.69而小于1时，就必须使用动态调度，且动态调度在调度过程中，任务的数量可变化．因此，动态调度有更宽的适用范围，对动态调度的分析是十分必要的．

实时调度在理论分析时，都假设系统能识别任意多的优先级．当实时调度应用于实际的任务系统时，仅能使用有限的优先级数量，在实际的任务系统中进行调度分析时，必须解决2个问题：1)确定任务系统调度所需的最小优先级数量；2)按照优先级数量，如何进行任务分组．文献[5-6]提出了一种系统能识别最少优先级的判断方法，但它们讨论的都是以RM算法为内容的静态调度；文献[7]在文献[6]的基础上，分析在动态调度的情况下，任务系统调度需要的最少优先级数量的判定方法；本文在文献[7]基础上，分析了任务系统分组的算法及最优分组的判定条件．

1 在有限优先级情况下的动态调度

考虑一个周期任务系统 $S(n)$，由 n 个可剥夺的相互独立的实时周期任务组成，表示为

$$\tau_1, \tau_2, \cdots, \tau_n, \tau_i = (a_i, C, T_i, D_i) \tag{1}$$

收稿日期：2007-03-13．
基金项目：国家自然科学基金项目(90407017)；北京市教委基金项目(KP2701200201)．
作者简介：何福贵(1966－)，男，山西忻州人，讲师；侯义斌(1952－)，男，陕西武功人，教授，博士生导师．

其中，a_i 表示任务的到达时间；C_i 表示要求的执行时间；T_i 表示周期；D_i 表示相对截止期，则 τ_i 的绝对截止期为 $d_i = a_i + D_i$，本文假设这些任务以绝对截止期不减的次序排列．为了分析方便，$\forall\ \tau_i$，$T_i = D_i$．

$$W(t) = \sum_{i=1}^{n} \left\lceil \frac{t}{T_i} \right\rceil \cdot C_i,\quad \lceil\ \rceil \text{表示向上取整．} \tag{2}$$

设 M 表示所有任务周期的最小公倍数．当 $W(M) < M$ 时，定义为任务系统不饱和，具有 $M \sim W(M)$ 个空槽；当 $W(M) = M$ 时，定义为任务系统饱和；当 $W(M) > M$ 时，任务系统不可调度．根据文献[5]有

$$e_{j(n)} = \min t \mid t = j + W_n(t) \tag{3}$$

式中 $e_{j(n)}$ 表示在有 n 个任务的系统中第 j 个空槽．

如果在系统中优先级的数量有限，且小于任务系统中的任务数量，则必须将任务系统中的任务进行分组．假设系统中能使用的优先级数量为 m，则分组数量$\leqslant m$，且每组至少包含 1 个任务．

设在某时刻 t，$S(n)$中 n 个任务同时到达，在 t 时刻之前没有任务执行，如果在 t 时刻之前有任务执行可归结到后面的情况，为分析方便，设 $t = 0$，那么这些任务的相对截止期就等于绝对截止期．这些任务可分为 m 组，表示为 $G(S(n)) = \{Q_1, Q_2, \cdots, Q_m\}$，$|Q_i|$ 表示 Q_i 组中任务的数量，则 $\sum_{i=1}^{m} |Q_i| = n$，且 $\forall\ i \in S(n)$，$\exists!\ Q_h \mid \tau_i \in Q_h$，其中 $\exists!$ 表示存在唯一的一个．

τ_i^j 表示第 j 组的第 i 个任务；C_i^j 表示第 j 组的第 i 个执行要求；D_i^j 表示第 j 组的第 i 个相对截止期．$\overline{Q}(j)$表示优先级高于第 j 组任务的集合，明显地，$C_1^2 = C_{|Q_1|+1}$，$D_1^2 = D_{|Q_1|+1}$，$\overline{Q}(j) = \bigcup_{g=1}^{j-1} Q_g$．

C^j 表示属于第 j 组中的所有任务执行 1 次要求的时间之和，$C^j = \sum_{h=1}^{|Q_j|} C_h^j$．

定理 1 给定任务集合 $S(n)$和任务组 $G(S(n))$，则 $G(S(n))$可使用 EDF 可调度，组内的任务可使用任意调度算法调度，如果下列条件成立：

$$\forall\ j = 1, 2, \cdots, m,\ D_1^j \geqslant \min t \mid t = C^j + W_{|\overline{Q}(j)|}(t) = e_{C^j(\overline{Q}(j))} \tag{4}$$

下面的定理说明在有限优先级的情况下，任务系统 $S(n)$可调度所需要的最少的优先级数量．

定理 2 给定任务系统 $S(n)$，使用 EDF 可调度，如果每组都是饱和的，那么这个划分 $G(S(n))$为最小数量的分组．

任务系统 $S(n)$中任务在某时刻 t 到达了 1 个新任务，在 t 时刻之前有 k 个任务到达，这 k 个任务是可调度的．但是在 t 时刻活动的任务有 j 个，$j \leqslant k$．活动的任务定义为：τ_i 是活动的，那么在 t 时刻 τ_i 的剩余执行时间$\geqslant 0$ 且 $t \leqslant d_i$．在时刻 t 到达了 1 个新任务，这些任务是 EDF 可调度的．

在 t 时刻活动的任务 $S(j)$为$(\tau_i, \tau_{i+1}, \cdots, \tau_{i+j})$，它们的绝对截止期为：$(d_i, d_{i+1}, \cdots, d_{i+j})$，其中 $d_i = a_i + D_i$，a_i 表示任务 τ_i 在 t 时刻之前最近一次的到达时间，在 t 时刻这些任务的剩余计算时间为：$(r_i, r_{i+1}, \cdots, r_{i+j})$，设在 t 时刻到达的任务为 τ_s，则有 $d_s = t + D_s$．

定理 3 如果新到达的任务为 τ_s，对于 $S(j)$有 $d_{i+l-1} \leqslant d_s \leqslant d_{i+l}$，$1 \leqslant l \leqslant j$，这些任务由 EDF 判定可调度，且 $C_s + r_{i+l} \leqslant d_s$，则 τ_s 的优先级可等于 τ_{i+l}的优先级(即 τ_s 和 τ_{i+l}可分为一组)，且任务系统可调度．

上面的定理给出了任务系统需要的最小优先级数量的判断方法．

2 任务系统最小分组的算法

设一任务组共有 10 个任务，任务的属性值见表 1．可调度的任务集合的最小分组算法的步骤如下：

1) 根据式(3)计算出任务组中各个任务的 $e_{C_i(i-1)}$，见表 1．

2) 根据定理 2 对任务进行最小分组，列出所有的分组．见表 2．

表 1　任务分组计算

Table 1　Tasks' grouping calculation

i	C_i	D_i	$e_{C_i(i-1)}$	i	C_i	D_i	$e_{C_i(i-1)}$
1	1	5	1	6	1	18	8
2	2	10	3	7	1	20	9
3	1	10	4	8	1	20	10
4	1	10	5	9	1	20	18
5	1	15	7	10	1	20	20

表 2　任务的原始分组

Table 2　Tasks' original grouping

i	原始分组	i	原始分组
1	$A=\{\tau_1,\tau_2,\tau_3,\tau_4\}$	6	$F=\{\tau_6,\tau_7,\tau_8,\tau_9\}$
2	$B=\{\tau_2,\tau_3,\tau_4,\tau_5,\tau_6,\tau_7,\tau_8\}$	7	$G=\{\tau_7,\tau_8,\tau_{10}\}$
3	$C=\{\tau_3,\tau_4,\tau_5,\tau_6,\tau_7,\tau_8\}$	8	$H=\{\tau_8,\tau_9,\tau_{10}\}$
4	$D=\{\tau_4,\tau_5,\tau_6,\tau_7,\tau_8\}$	9	$I=\{\tau_9,\tau_{10}\}$
5	$E=\{\tau_5,\tau_6,\tau_7,\tau_8\}$	10	$J=\{\tau_{10}\}$

3) 去掉表 2 中的子集部分. 对于表 2, E、D、C 为 B 的子集, J、I、H 为 G 的子集. 去掉子集后见表 3.

表 3　去掉子集后的分组

Table 3　Independent grouping

i	分组	i	分组
1	$A=\{\tau_1,\tau_2,\tau_3,\tau_4\}$	3	$F=\{\tau_6,\tau_7,\tau_8,\tau_9\}$
2	$B=\{\tau_2,\tau_3,\tau_4,\tau_5,\tau_6,\tau_7,\tau_8\}$	4	$G=\{\tau_7,\tau_8,\tau_9,\tau_{10}\}$

4) 建立表 4, 表中的第 1 列为去掉子集后的原始分组, 后面的每 1 列对应任务集合中每个任务. 在每个交叉部分, 如果这个分组包含这个任务, 画 1 个对号.

5) 表中有一些列只包含 1 个对号, 对应的这个任务将形成 1 个单独的分组. 这些分组组成表 5, 表 5 的每个原始分组至少其中 1 列只有 1 个对号.

表 4　任务分组表

Table 4　Tasks grouping

原始分组	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7	τ_8	τ_9	τ_{10}
A	√	√	√	√						
B		√	√	√	√	√	√	√		
F						√	√	√	√	
G							√	√	√	√

表 5　包含单独任务

Table 5　Containing single task

原始分组	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7	τ_8	τ_9	τ_{10}
A	√	√	√	√						
B		√	√	√	√	√	√	√		
G							√	√	√	√

6) 由表 5 可产生分组的划分, 分组的划分应根据列中只有 1 个对号的任务成为单独的一组, 有 2 个对号的列对应的任务可进入前一组, 也可进入后一组, 所有可能的分组见表 6.

表 6　所有可能的分组情况

Table 6　All grouping

编号	分组	编号	分组
1	$\{\tau_1\},\{\tau_2,\tau_3,\tau_4,\tau_5,\tau_6,\tau_7,\tau_8\},\{\tau_9,\tau_{10}\}$	7	$\{\tau_1,\tau_2,\tau_3\},\{\tau_4,\tau_5,\tau_6,\tau_7\},\{\tau_8,\tau_9,\tau_{10}\}$
2	$\{\tau_1,\tau_2,\},\{\tau_3,\tau_4,\tau_5,\tau_6,\tau_7,\tau_8\},\{\tau_9,\tau_{10}\}$	8	$\{\tau_1,\tau_2,\tau_3,\tau_4\},\{\tau_5,\tau_6,\tau_7\},\{\tau_8,\tau_9,\tau_{10}\}$
3	$\{\tau_1,\tau_2,\tau_3\},\{\tau_4,\tau_5,\tau_6,\tau_7,\tau_8,\tau_9\},\{\tau_9,\tau_{10}\}$	9	$\{\tau_1\},\{\tau_2,\tau_3,\tau_4,\tau_5,\tau_6\},\{\tau_7,\tau_8,\tau_9,\tau_{10}\}$
4	$\{\tau_1,\tau_2,\tau_3,\tau_4\},\{\tau_5,\tau_6,\tau_7,\tau_8\},\{\tau_9,\tau_{10}\}$	10	$\{\tau_1,\tau_2\},\{\tau_3,\tau_4,\tau_5,\tau_6\},\{\tau_7,\tau_8,\tau_9,\tau_{10}\}$
5	$\{\tau_1\},\{\tau_2,\tau_3,\tau_4,\tau_5,\tau_6,\tau_7\}\{\tau_8,\tau_9,\tau_{10}\}$	11	$\{\tau_1,\tau_2,\tau_3\},\{\tau_4,\tau_5,\tau_6\},\{\tau_7,\tau_8,\tau_9,\tau_{10}\}$
6	$\{\tau_1,\tau_2\},\{\tau_3,\tau_4,\tau_5,\tau_6,\tau_7\},\{\tau_8,\tau_9,\tau_{10}\}$	12	$\{\tau_1,\tau_2,\tau_3,\tau_4\},\{\tau_5,\tau_6\},\{\tau_7,\tau_8,\tau_9,\tau_{10}\}$

3 在分组中选择最优的分组

由任务系统分组算法可得到任务分组的集合，一般地，分组情况都大于1种，那么如何确定分组中哪个分组是最优的分组，根据文献[6,8]，在有限优先级和使用静态优先级算法情况下的判断条件是当第 i 个任务加入所有优先级比 i 高的任务中时，这些任务的最小执行要求为：$S_i=\min[W_i(t)/t]$，$\forall t|1\leqslant t\leqslant D_i$，这个公式是测量系统中 i 个任务的饱和度，这个值表示在$[1,D_i]$间隔内的最小执行要求．文献[8]证明了在有限优先级和任务分组条件下任务的最小执行要求，在文献[8]的基础上，可得出在动态调度时，任务的最小执行要求为

$$\forall \tau_i\in Q_j,\ S_i'=\min_{0<t\leqslant D_i}\left[\frac{1}{t}\left(C^j+\sum_{\forall h|h\in Q(j)}W_h(t)\right)\right] \tag{5}$$

根据式(5)，定义调度饱和度为：$S_{\max}=\max_{1\leqslant i\leqslant n}S_i$ 和 $S'_{\max}=\max_{1\leqslant i\leqslant n}S_i'$．这2个值表示在上述2种情况下可调度的临界点，这个值越小，饱和度越低，表示更好的调度性．定义：$R_S=S_{\max}/S'_{\max}$，它是2种情况下系统任务饱和度的比，这2种情况分别是在有限优先级的情况下任务没有分组和任务分组，R_S 的值应小于1，那么最好的分组就是最小的 R_S．在上面的例子中 $R_S=1$，所以所有的分组都相同．

4 结论

静态调度的调度开销较小，但它的适用范围小，而且要求在调度前已知任务系统中任务的参数值，当任务系统中的任务较多时，CPU的利用率仅为0.69以下；动态调度的调度开销较大，但它的CPU利用率高，对任务系统的限制较小，且在调度的过程中，任务的数量可变化，它的适应性更强．对于实时周期独立的可剥夺的任务集合，它们执行在一个单处理器上，如果运行在实际系统中，一般都有优先级数量的限制．现有文献描述的是在静态调度使用RM算法的情况下，任务分组和任务最小分组的方法．本文给出在使用EDF动态调度的情况下，任务分组和任务最小分组的方法，描述了分组算法详细的步骤和过程，可用于实际的任务系统中．

参考文献：

[1] LIU C L, LAYLAND J M. Scheduling algorithms for multiprogramming in a hard real-time environment[J]. Journal of the ACM, 1973, 20(1): 46-61.

[2] LEUNG J, WHITEHEAD J. On the complexity of fixed-priority scheduling of periodic, real-time tasks[J]. Performance Evaluation, 1982, 2: 237-250.

[3] DERTOUZOS M L, MOK A K. Multiprocessor on-line scheduling of hard-real-time tasks[J]. IEEE Trans on Software Engineering, 1989, 15(12): 1497-1506.

[4] SPRUNT B, SHA L, LEHOCZKY J. Aperiodic task scheduling for hard-real-time systems[J]. Real Time Systems, 1989, 1(1): 27-60.

[5] LEHOCZKY J P, SHA L. Performance of real-time bus scheduling algorithms[J]. ACM SIGMETRICS Performance Evaluation Review, 1986, 14(1): 44-53.

[6] OROZCO J, CAYSSIALS R, SANTOS J, et al. On the minimum number of priority levels required for the rate monotonic scheduling of real-time systems[C]// Proceedings of the 10th Euromicro Workshop on Real Time Systems. Los Alamitos, CA: IEEE Computer Society Press, 1998: 19-21.

[7] 何福贵，王家礼．基于有限优先级的动态调度分析[J]．电子科技大学学报(自然科学版)，2007，36(3)：545-547.
HE Fu-gui, WANG Jia-li. Dynamic scheduling algorithm based on limited number of priority[J]. Journal of University of Electronic Science and Technology of China(Natural Science Edition), 2007, 36(3): 545-547. (in Chinese)

[8] SPRUNT B. A periodic task scheduling for real-time systems[D]. Pittsburgh: Department of Electrical and Computer Engineering, Carnegie Mellon University, 1990: 106-116.

Dynamic Scheduling Grouping Algorithm Based on Limited Priority Levels

HE Fu-gui, HOU Yi-bin, LI Hui
(Embedded Software and Systems Institute, College of Software Engineering, Beijing University of Technology, Beijing 100022, China)

Abstract: Real-time scheduling is a key in real-time system, and real-time dynamic scheduling is the main part in real-time scheduling. When carry out theoretic analysis of real-time scheduling, real-time theory assume that system owns enough number of priority levels. But in practical tasks, system only limited priority levels could be used. In practical tasks system, analysis for dynamic scheduling included two continuous parts: judged method that tasks systems require minimum number of priority levels for dynamic scheduling and grouping algorithm for tasks system. On the basis of previous result, the paper presented grouping algorithm for tasks system and judged condition for best grouping. The steps and the process for the algorithm are described in detail.

Key words: dynamic scheduling; limited priority levels; tasks grouping algorithm; real-time scheduling

(责任编辑　苗艳玲)

(上接第 855 页)

Flexural Toughness of Cellulose and Hybrid Fiber Reinforced Concrete Beams

DENG Zong-cai, ZHANG Peng-fei, XUE Hui-qing, LI Peng-yuan, SHE Xiang-jun
(College of Architecture and Civil Engineering, Beijing University of Technology, Beijing 100022, China)

Abstract: The cellulose fiber (ultra-fiber 500) is a new type natural cellulose fiber. In order to study the flexural toughness of cellulose fiber reinforced concrete, the flexural toughness experiments of concrete with synthetic, steel fibers, and hybrid fiber are made at the same time, and the load-deflection curve of all beams are obtained. Based on the ASTM method, the flexural toughness of concrete with cellulose, synthetic, steel fibers and hybrid fibers are all analyzed. The experimental results indicate that the cellulose fiber can improve the flexural toughness and the deformation ability of concrete beams. The flexural toughness index I_5 and I_{10} of concrete with cellulose fiber are 3.0 and 5.8 times higher than that of the plain concrete respectively. For the same ratio of volume of fibers, the flexural toughness index of cellulose fiber reinforced concrete is higher than that of concrete with PP fiber. Cellulose and steel hybrid fibers can improve the toughness and deformation ability of concrete remarkably, and the failure mode changes from brittle to ductile fracture.

Key words: cellulose fiber; steel fiber; hybrid fiber; flexural toughness; concrete

(责任编辑　苗艳玲)