提高软非周期任务响应性能的调度算法*

何军 孙玉方

(中国科学院软件研究所 北京 100080)

摘要 实时环境中常常既包含硬周期任务,又包含软非周期任务,引入一种改进软非周期实时任务响应时间的算法.已有的解决混合任务调度问题的方法都是基于速率单调(Rate Monotonic)策略的,其中从周期任务"挪用时间"的算法被证明优于其他所有算法.但是,速率单调算法限制了处理器的使用率,从而使周期任务的可"挪用"时间受到限制.最后期限驱动(Deadline Driven)策略DD可使潜在的处理器利用率达到100%.新算法正是在周期任务的调度中适当加入了DD策略,从而使非周期任务的响应时间得以缩短.仿真实验的结果表明,这种算法的性能优于已有的所有算法,而由它所带来的额外开销却不算很高.

关键词 实时调度,周期任务调度,非周期任务调度,速率单调算法,最后期限驱动算法.

中图法分类号 TP316

1973年,C. L. Liu 和 J. W. Layland 提出了两种调度硬周期实时任务的算法[1],一种采用固定优先级方案,称为速率单调(Rate Monotonic)算法,简称RM算法;另一种采用动态优先级方案,称为最后期限驱动(Deadline Driven)算法,简称DD算法.前者按任务的周期长短分配优先级,周期越短,优先级越高;后者动态分配优先级,任一时刻,最后期限近的任务优先级高.文献[1]中还证明,RM算法是固定优先级方案中的最优算法,它可使周期任务负载的最小上确界(Least Upper Bound)达到 $n(2^{1/n}-1)$(n 是周期任务数);DD算法是动态优先级方案中的最优算法,它可使周期任务负载的最小上确界达到1.0.在文献[1]发表之后,RM算法由于其简单性而得到了广泛应用.人们在它的基础上提出了许多算法,用于解决各种实时调度问题,如实时任务同步、瞬间超载(实时任务要求的计算量超过可用的CPU时间)以及周期任务和非周期任务的混合调度问题等.周期、非周期任务混合调度问题之所以重要,是因为许多实时系统中都既包含硬周期任务又包含软非周期任务.解决这个问题的出发点是在保证不影响周期任务调度的同时,尽早完成非周期任务.

传统上解决混合任务调度问题的方法有两种:①后台执行法,即以后台任务的形式执行非周期任务,这种方法易于实现,但会导致非周期任务的响应时间很长;②轮询法,即用一个高优先级的轮询任务(或称服务器)周期性地为非周期任务服务.轮询法大大优于后台执行法,因为服务器每过一个固定时间间隔,就为非周期任务提供了一段高优先级的可用执行时间(我们称之为带宽).但是,如果服务器周期开始时没有非周期任务到达,带宽将被丢弃,因此,若一个非周期任务恰在带宽丢弃后到达,则它将或者等到下一个轮询周期开始时才能运行,或者就在后台执行.

为了克服轮询法的缺陷,J. P. Lehoczky 等人提出了优先级交换PE(priority exchange)算法和可延期服务器DS(deferrable server)算法[2],Sprunt 稍后又提出了间发服务器SS(sporadic server)算法.[3]PE,DS和SS三种方法被归为带宽保留算法,当服务器周期开始而没有非周期任务时,它们采用了各自不同的方法来保留非周期服务器的带宽.在某些情况下,这些算法可以达到很理想的效果,但在周期任务负载较大、非周期任务所需执行时间较多等情况下,它们的性能会急剧下降.

近年来,Lehoczky 等人提出了另一种解决混合任务调度问题的方案[4],本文简称它为OFP算法,O表示最优(Optimal),FP表示固定优先级(Fixed-Priority).与轮询法,PE,DS和SS等方法不同,OFP算法不使用服务器为非周期任务服务,而是在出现非周期任务时,以满足周期任务的最后期限为前提,尽可能地推迟周期任务的执行,从而"挪出"一段时间供非周期任务使用,我们将最多可挪用的时间称为可延缓时间.

OFP算法事先算出在0到超周期(Hyperperiod,是所有周期任务周期的最小公倍数)时间内,每个周期任务的每

---

* 作者何军,女,1969年生,博士,助理研究员,主要研究领域为unix操作系统,实时调度算法.孙玉方,1947年生,研究员,博导,主要研究领域为操作系统,中文信息处理,数据库.

本文通讯联系人:孙玉方,北京 100080,中国科学院软件研究所

本文1997-01-09收到原稿,1997-09-26收到修改稿

个作业有多少可延缓(或可挪用)时间,并将这些值保存在一张表(可延缓时间表)中.在运行过程中,每个周期任务需要一个计数器,用于跟踪记录已处理的优先级高于或等于这个任务的周期作业量,另外还需一个计数器用于记录从0时刻到当前时刻非周期任务使用的总时间量.当非周期任务需要执行时间时,根据延缓时间表和非周期任务已占用的时间量等数值来计算当前最多可将所有周期任务延迟执行多少时间,并将这段时间分配给非周期任务.

OFP可使非周期任务的响应时间达到非常好的效果,它还可扩展,用于解决硬非周期任务调度问题.但它也存在一些缺点:(1) 当超周期很大时,周期任务的可延缓时间表开销也会很大,而且即使在无非周期任务时,还需跟踪记录周期和非周期任务已完成的作业量,因此造成了不必要的运行开销;(2) OFP是基于固定优先级的,文献[1]中指出,固定优先级方案下处理器的利用率要受到一定限制,因此,周期任务的可延缓时间也会受到限制.

DD算法的潜在处理器利用率可达到1,我们不难想到,当出现非周期请求时,如果在未来的一定时间内对周期任务的调度使用DD策略,那么周期任务的可延缓时间将大于或等于完全使用RM策略时得到的值,由此使非周期任务的响应时间得以改善.

本文引入了一种新的算法,称为加入DD策略的软非周期任务调度算法,与OFP相对应,简称为ODD算法.ODD算法事先并不计算每个周期任务的每个作业可"被挪用"多少时间.当非周期任务请求执行时,它首先考虑,在使用RM策略的情况下,若将这个非周期任务执行完,是否会影响周期任务的执行,如果不会,则可将非周期任务执行完再按RM策略调度周期任务;如果会,则计算当在未来的某个时间段内使用DD策略调度周期任务时,最多可挪用出多少时间供非周期任务使用.ODD算法与OFP算法相比,减少了维护和操作可延缓时间表的开销,在处理非周期任务时,虽然稍稍增加了计算量,但它可使非周期任务的响应时间近乎达到理想化.

本文第1节介绍ODD算法并举例进行说明;第2节介绍模拟ODD算法的实验结果;最后是小结.

1 ODD算法

1.1 任务模型和假设

为了便于分析,我们建立了以下任务模型:假设实时系统中有n个按优先级从高到低排序的周期任务$\tau_1,\tau_2,\ldots,\tau_n$,其中$\tau_1$的周期最短,$\tau_n$的周期最长.任务$\tau_i$的最坏执行时间是$C_i$,周期为$T_i$,最后期限与$T_i$相等.与非周期任务相关的参数是到达时间和请求执行时间.我们还作了以下假设:

假设1. 每个任务新一轮的请求均在这个任务周期开始时到达;

假设2. 优先级高的任务可抢占优先级低的任务;

假设3. 任务不被挂起;

假设4. 任务切换时的开销忽略不计;

假设5. 非周期任务按先来先服务的顺序处理.

1.2 算法原理及其说明

我们的目标是使非周期响应时间尽可能短,因此任一时刻,如果有非周期任务需要执行,我们就要确定从这个时刻起最多可以用多少时间连续运行非周期任务,同时又不影响周期任务的执行.我们的分析以Lehoczky在文献[5]中所作的任务可调度性分析为基础.假定当前时刻为t,非周期任务从t运行到t_2,如果周期任务的优先级按RM策略排定,那么对$i=1,2,\ldots,n$,下列不等式必须满足,否则周期任务将会错过最后期限.

$$P_i(t,D_i(t)) = \sum_{j=1}^{i} RC_j(t) + \sum_{(j>i,D_j(t)\leqslant D_i(t))} RC_j(t) + \sum_{j=1}^{i-1} C_j \cdot \lceil \max(0, D_i(t) - D_j(t))/T_j \rceil \leqslant D_i(t) - (t_2 - t) \tag{1}$$

上式中,$D_i(t)$表示t时刻τ_i所在周期的结束时间,$D_i(t)=\lceil t/T_i \rceil \cdot T_i$;$RC_i(t)$表示$\tau_i$在其当前周期还需执行的时间;$P_i(t,D_i(t))$表示为满足所有周期任务在$[t,D_i(t)]$内的最后期限而需完成的执行时间.

如果对于$i=1,2,\ldots,n$中的任一值,式(1)不能成立,则说明若按RM策略调度周期任务,非周期任务不能从t运行到t_2.对公式(1),我们也可以这样理解:在使用RM策略的前提下,为了使所有周期任务都能满足最后期限要求,从t时刻开始,最多可以将所有周期任务延迟执行$\min(D_i(t)-P_i(t,D_i(t)))$个时间单位,如果下式成立,则非周期任务可以占用$[t,t_2]$时间段.

$$t_2 - t \leqslant \min(D_i(t) - P_i(t,D_i(t))). \tag{2}$$

如果用DD策略调度周期任务,那么将所有周期任务从t延迟到t_2,应满足以下公式.

$$P_i'(t,D_i(t))=\sum_{(1\leqslant j\leqslant n,D_j(t)\leqslant D_i(t))}RC_j(t)+\sum_{j=1}^{i}C_j\cdot\lfloor\max(0,D_i(t)-D_j(t))/T_j\rfloor \leqslant D_i(t)-(t_2-t) \tag{3}$$

$P_i'(t,D_i(t))$表示当使用DD策略时,为满足所有周期任务在$[t,D_i(t)]$内的最后期限而需完成的执行时间.显然,当$j>i$时,由于$T_j>T_i$,因此有$\lfloor\max(0,D_i(t)-D_j(t))\rfloor=0$,继而有$P_i'(t,D_i(t))\leqslant P_i(t,D_i(t))$,即在$[t,D_i(t)]$时间内使用DD策略可获得的延缓时间大于或等于用RM策略所获得的延缓时间.

正是基于以上分析,我们想到要在周期任务的调度中适当使用DD策略,使非周期任务挪用到更多执行时间,从而缩短非周期响应时间.由于DD策略的运行时间开销大于RM策略,因此,在ODD算法中,我们在利用DD策略以获得最大可延缓时间的同时,还应确定可结束DD策略并恢复RM策略的最早时刻.最早恢复时间的确定方法在下节中给出.

1.3　算法描述

假设所有周期任务都从0时刻开始请求执行,从这个时刻起到第1个非周期请求出现为止,所有周期任务按RM策略调度.当t时刻非周期任务J_k请求执行α_k个时间单位时,所有任务(包括周期的和非周期的)将按以下策略进行调度:

(1) 对所有周期任务计算$D_i(t)$和$P_i(t,D_i(t))(i=1,2,\cdots,n)$;

(2) 若对所有$P_i(t,D_i(t))$,公式(4)成立,则J_k可以从t时刻执行到$t+\alpha_k$时刻,其后,所有周期任务仍按RM策略调度,否则转入第3步:

$$t+\alpha_k+P_i(t,D_i(t))\leqslant D_i(t) \tag{4}$$

(3) 将不满足式(4)的任务形成集合T',对T'中的任务τ_i计算$P_i'(t,D_i(t))$和$D_i(t)-P_i'(t,D_i(t))$,并按下式取非周期任务J_k的可执行时间α_k':

$$\alpha_k'=\min(D_i(t)-t-P_i'(t,D_i(t)),\alpha_k)\quad(\tau_i\in T') \tag{5}$$

对于$\tau_m\in T'$应有$\alpha_k'\leqslant Dm(t)-t-P_m'(t,D_m(t))$,但是将$\alpha_k$替换为$\alpha_k'$后,$T'$中仍可能有不符合式(4)的任务,因此,结束使用DD策略的时间应取:

$$t_{DD}=\max_{(\tau_m\in T',t+\alpha_k'+P_m'(t,D_m(t))>D_m(t))}(t+\alpha_k'+P_m'(t,D_m(t))) \tag{6}$$

若在$[t+\alpha_k',t_{DD}]$时间段内不再有非周期任务需要执行,则从t_{DD}开始周期任务的调度将恢复使用RM策略,否则,仍需按照上述步骤重新确立调度方案.

1.4　ODD算法举例

本节通过一个例子说明ODD算法的调度过程.考虑两个周期任务τ_1和τ_2,其中$C_1=1,T_1=D_1=10,C_2=1,T_2=D_2=14$,非周期任务$J_1$在$t=14$到达,需执行$\alpha_1=13$个时间单位.在$t=0$到$t=14$之间,$\tau_1$和$\tau_2$按RM策略调度.当$t=14$时,$J_1$到达,此时

$$D_1(t)=20,D_2(t)=28,RC_1(t)=0,RC_2(t)=1;$$
$$P_1(t,D_1(t))=RC_1(t)=0;$$
$$P_2(t,D_2(t))=RC_1(t)+RC_2(t)+\lceil(D_2(t)-D_1(t))/T_1\rceil\cdot C_1=2;$$
$$P_1(t,D_1(t))+\alpha_1+t=27>D_1(t)=20;$$
$$P_2(t,D_2(t))+\alpha_1+t=29>D_2(t)=28.$$

上面这些算式表明,若用RM策略调度τ_1和τ_2,J_1不能执行13个时间单位.那么,再看看用DD策略的结果.根据式(3)可得

$$P_1'(t,D_1(t))=0,P_2'(t,D_2(t))=1,$$

根据式(5)、(6),有

$$D_1(t)-t-P_1'(t,D_1(t))=6,$$
$$D_2(t)-t-P_2'(t,D_2(t))=13,$$
$$\alpha_1'=\min(6,13,\alpha_1)=6,$$
$$t_{DD}=t+\alpha_1'+P_1'(t,D_1(t))=20.$$

因此,J_1可以在$t=14$和$t=20$之间运行.$t=20$时,J_1仍需执行7个时间单位,这时再计算公式(1),可得

$$D_1(t)=30,D_2(t)=28,RC_1(t)=1,RC_2(t)=1;$$

$$P_1(t,D_1(t)) = RC_1(t) = 1;$$
$$P_2(t,D_2(t)) = RC_1(t) + RC_2(t) = 2;$$
$$P_1(t,D_1(t)) + 7 + t = 28 < D_1(t) = 30;$$
$$P_2(t,D_2(t)) + 7 + t = 29 > D_2(t) = 28.$$

上式说明，若用RM策略调度τ_1和τ_2，J_1不能执行7个时间单位(仍记为α_1)，这时$T'=\{\tau_2\}$，因此，只需对τ_2计算$P_2'(t,D_2(t))$.

$$P_2'(t,D_2(t)) = 1.$$

根据式(6)得

$$D_2(t) - t - P_2'(t,D_2(t)) = 28 - 20 - 1 = 7,$$
$$\alpha_1' = \min(7,\alpha_1) = 7,$$
$$t_{DD} = t + \alpha_1' + P_2'(t,D_2(t)) = 28.$$

上式说明，J_1可以从$t=20$起，执行7个时间单位，在[27,28]时间内需使用DD策略调度τ_1和τ_2. 至$t=27$时，J_1已全部完成，响应时间为13，而使用DD策略的时间只有1个时间单位.

2 仿真实验

为了比较各种非周期任务调度算法的性能，我们做了一些实验来仿真任务的调度过程，本节给出一些实验结果. 由于文献[2,3]中已指出DS,PE,SS算法的性能优于后台执行法和轮询法，文献[4]中则表明OFP算法又比DS,PE,SS算法优越，因此，在本节中，我们只显示了ODD算法和OFP算法的比较结果.

2.1 实验描述

在本实验中，我们采用了与文献[3]中仿真实验类似的仿真条件. 我们随机产生了由9个周期任务组成的周期任务集. 为了比较在各种周期负载下各个算法的性能，对这组任务选择了3个周期负载，70%、80%和90%用于实验，70%略低于RM策略下周期负载的最小上确界$9(2^{1/9}-1)\approx 72\%$，80%略高于此值，90%代表周期负载较高的情况. 表1给出了周期任务的周期和对应每个周期负载的执行时间.

非周期任务这样选择：非周期任务的到达时间和执行时间均符合泊松分布(Poisson Distribution)，平均到达时间取周期任务1的周期的50%、100%和200%(即53、105和210). 这种取值是为了显示非周期任务的到达频率从高到低时，各种算法的性能比较. 对每种周期负载和非周期任务到达时间的组合，我们又选取了5种非周期负载，使得总负载(周期负载+非周期负载)的范围从略高于周期负载到99%，对每组实验，我们均产生了100个非周期任务用于调度仿真实验.

表1 仿真实验中的周期任务集

	周期	执行时间		
		70%	80%	90%
1	105	9	10	12
2	120	16	18	20
3	126	4	4	6
4	140	8	10	11
5	280	21	24	27
6	420	21	24	27
7	630	76	87	98
8	840	109	125	141
9	2 520	13	15	17

2.2 平均响应时间的实验结果

为了显示OFP和ODD算法的非周期响应性能，图1～3分别给出在70%、80%和90%的周期负载下，这些算法的平均响应时间. 其中每个小图的x轴表示任务的总负载量，y轴表示平均非周期响应时间. 每个小图中还给出了一条M/M/1曲线，表示在没有周期任务的情况下，可获得的理想非周期响应时间.

从图1～3我们可以看出，在各种情况下，ODD算法都优于OFP算法，周期负载越大，这种优越性越明显. 当周期负载较低(70%)时，小图中的3条曲线都很接近(如图1所示)，随着周期负载的增加，OFP算法曲线与M/M/1曲线

的距离渐渐拉开,而 ODD 算法曲线仍趋近于 M/M/1 曲线.尤其是当周期负载达到 90%时,OFP 算法的平均非周期响应时间几乎都超过了理想响应时间的 2 倍,而 ODD 算法的平均非周期响应时间曲线却几乎与 M/M/1 曲线重合.这也说明,当周期负载大而非周期负载较小时,ODD 表现出的性能最佳.

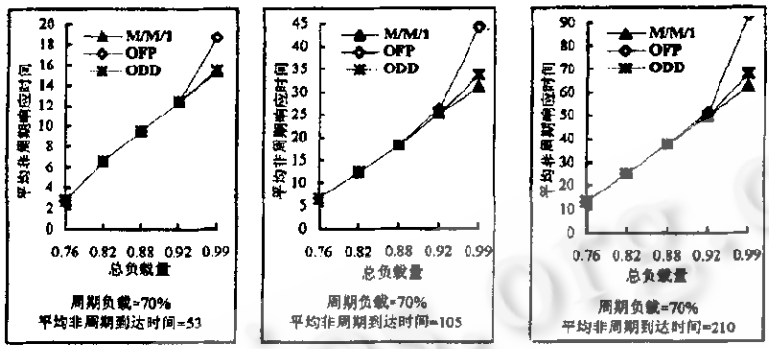

图 1　周期负载为 70%时的平均非周期响应时间

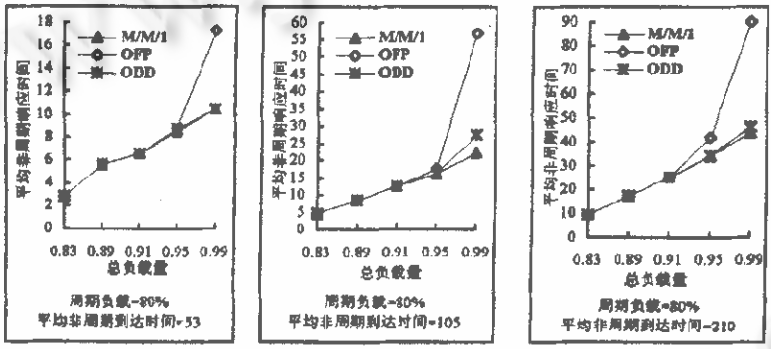

图 2　周期负载为 80%时的平均非周期响应时间

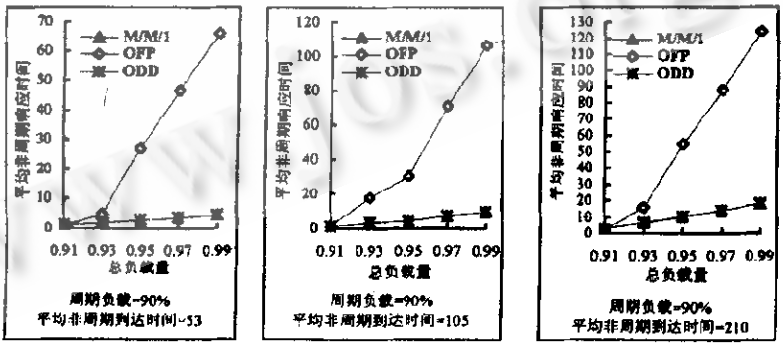

图 3　周期负载为 90%时的平均非周期响应时间

我们还发现,当非周期任务到达频率较高(见各图中的第 1 个小图)时,ODD 算法曲线也几乎与 M/M/1 曲线重合.这是因为,非周期任务虽然来得频繁,但每个作业都较小,因此每次需要的可延缓时间也较少.

本文 1.2 节曾指出,由于 DD 策略的运行时间开销大于 RM 策略,因此在整个时间轴上,使用 DD 策略的时间所占百分比也就成了衡量 ODD 算法所导致的额外开销的一个重要参数.在仿真实验中,我们还统计了使用 DD 策略的

时间与总时间的比率，总时间以完成最后一个非周期任务时，周期任务所处的超周期结束为止. 表2给出各种组合下的统计结果.

表2　ODD算法中使用DD策略的时间比　　(%)

周期负载 / 总负载 / 平均非周期到达时间(s)	70%					80%					90%				
	76%	82%	88%	92%	99%	83%	87%	91%	95%	99%	91%	93%	95%	97%	99%
53	0	0	0	0	1.6	0	0	0	0	2.14	0	1.28	5.86	8.4	11.54
105	0	0	0	0.53	4.1	0	0	0	0.67	8.47	0	3.38	5.94	15.25	21.09
210	0	0	0	0.33	6.73	0	0	0	1.64	9.43	0	1.94	8.84	14.25	18.91

从表2可以看出，在ODD算法中，大多数情况下不需使用DD策略，若使用DD策略，则时间比率最坏情况不超过1/4，平均情况约为1/20，最好情况还不到1/250，因此，ODD策略所增加的运行时间负载不会太高.

3　OFP和ODD算法与带宽保留算法的比较

OFP和ODD算法在非周期任务响应性能上明显优于带宽保留算法，但好的响应性能也需付出一定的代价：与带宽保留算法相比，它们的复杂度显然是要高一些的，另外，它们的文境切换开销也会有所增加.

在前面的讨论中，我们均未考虑文境切换的问题，实际上，任务发生抢占就会有文境切换的问题，抢占次数越多，文境切换次数也越多，切换开销也越大.

OFP和ODD算法的文境切换开销大于带宽保留算法，这是因为，在这两种算法中，每当非周期任务到达，只要周期任务有可延缓时间，非周期任务总可以抢占周期任务(只要周期任务正在运行)；而在带宽保留算法中，当非周期任务到达时，如果无服务器能量，它就不可能抢占周期任务的执行，尽管这时周期任务可能有延缓时间. 也就是说，OFP和ODD算法中非周期任务抢占周期任务的比率较高，因此，它们的文境切换开销大于带宽保留算法. OFP算法和ODD算法原理相似，都是从周期任务挪用时间为非周期任务服务，因此，它们的文境切换开销应基本相同. 我们做的一些仿真实验证实了上述想法，由于篇幅所限，本文不再给出仿真实验结果.

4　结　论

本文介绍了一种在硬周期环境中改进软非周期任务响应时间的方法，称为ODD算法. 它的主要思想是，在有非周期请求的情况下将周期任务延迟执行，即从周期任务"挪用"时间. 传统的混合任务调度算法都是基于RM策略的，ODD算法除使用RM策略外，在周期任务的调度中适当加入了动态的DD策略，从而使周期任务的可延迟时间加大，并因此而改进了非周期响应时间. ODD算法与OFP算法相比，去除了可延缓时间表的内存开销，也不必记录周期、非周期任务完成的作业量，但它在处理非周期任务时稍稍增加了计算量. 仿真实验表明，ODD算法的非周期任务响应性能优于已有的所有算法，并且周期任务的负载越大，优越性越明显. 但好的响应性能也需付出一定的代价：与带宽保留算法相比，它增加了一定的文境切换开销. 实验结果还表明，使用动态策略的时间比率并不很高，在大多数情况下，周期任务的调度还是基于RM策略的，因此增加的运行时的负载不会很大.

参考文献

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

2　Lehoczky J P, Sha L, Strosnider J K. Enhanced aperiodic responsiveness in hard real-time environments. In: Proceedings of IEEE Real-Time System Symposium. December 1987. 261～270

3　Sprunt B. Aperiodic task scheduling for real-time systems [Ph.D. Thesis]. Carnegie Mellon University, 1990

4　Lehoczky J P, Thuel S R. An optimal algorithm for scheduling soft-aperiodic tasks in fixed-priority preemptive systems. In: Proceedings of IEEE Real-Time System Symposium. December 1992. 110～123

5　Lehoczky J P, Sha L, Ding Y. The rate-monotonic scheduling algorithm: exact characterization and average case behavior. In: Proceedings of IEEE Real-Time System Symposium. December 1989. 166～171

An Algorithm to Improve the Responsive Performance for Scheduling Soft-Aperiodic Tasks

HE Jun　SUN Yu-fang

(*Institute of Software　The Chinese Academy of Sciences　Beijing*　100080)

Abstract　Many real-time systems have hard deadline periodic tasks along with soft deadline aperiodic tasks. An algorithm to improve response time of soft aperiodic tasks is presented in this paper. Existing methods of scheduling both periodic and aperiodic tasks are all based on RM (rate monotonic) scheme. Among them, stealing slack time from periodic tasks has been proved to be an optimal method. However, the RM scheme limits the processor utilization, therefore limits the stealable time of periodic tasks. The potential processor utilization of the DD (deadline driven) scheme can reach 100%. The presented algorithm applies the DD scheme to periodic task scheduling as it needs, and shortens the response time of aperiodic tasks. The experimental results show that the new algorithm provides performance higher than all existing algorithms, and the increased run-time overhead is not very high.

Key words　Real-time scheduling, periodic task scheduling, aperiodic task scheduling, rate monotonic algorithm, deadline driven algorithm.